IBM® Information Management Software

Security Functions of IBM DB2 10 for z/OS

Implement separation of duties

Audit application and system activity

Protect from intrusions and misplacements

Paolo Bruni
Marcelo Antonelli
Hyun Baek
Rick Butler
Ernie Mancill

Redbooks

ibm.com/redbooks

International Technical Support Organization

Security Functions of IBM DB2 10 for z/OS

September 2011

SG24-7959-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xxi.

First Edition (September 2011)

This edition applies to IBM DB2 Version 10.1 for z/OS (program number 5605-DB2).

Contents

Figures

Tables

Examples

Notices

Trademarks

Preface

IBM® DB2® 9 and 10 for z/OS® have added functions in the areas of security, regulatory compliance, and audit capability that provide solutions for the most compelling requirements.

DB2 10 enhances the DB2 9 role-based security with additional administrative and other finer-grained authorities and privileges. This authority granularity helps separate administration and data access that provide only the minimum appropriate authority.

The authority profiles provide better separation of duties while limiting or eliminating blanket authority over all aspects of a table and its data. In addition, DB2 10 provides a set of criteria for auditing for the possible abuse and overlapping of authorities within a system.

In DB2 10, improvements to security and regulatory compliance focus on data retention and protecting sensitive data from privileged users and administrators. Improvements also help to separate security administration from database administration.

DB2 10 also lets administrators enable security on a particular column or particular row in the database complementing the privilege model.

This IBM Redbooks® publication provides a detailed description of DB2 10 security functions from the implementation and usage point of view. It is intended to be used by database, audit, and security administrators.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Paolo Bruni is an information management software Project Leader at the International Technical Support Organization, based in the Silicon Valley Lab. He has authored several IBM Redbooks publications about DB2 for z/OS and related tools, and has conducted workshops and seminars worldwide. During Paolo's years with IBM, in development and in the field, his work has been mostly related to database systems.

Marcelo Antonelli is a DB2 Specialist with IBM Brazil. He has 20 years of experience working with DB2 for z/OS. Marcelo holds a graduate degree in System Analysis from PUCC in Campinas, Sao Paulo. His areas of expertise include database design, system administration, and performance. Marcelo is currently supporting an outsourcing internal IBM account from EUA, as well as assisting IBM technical professionals working with DB2. Marcelo co-authored the books *DB2 for z/OS and OS/390DB2 9 for z/OS: Using the Utilities Suite*, SG24-6289, *IBM DB2 Performance Expert for z/OS Version 2*, SG24-6867, *Administration Solutions for DB2 UDB for z/OS*, SG24-6685, *A Deep Blue View of DB2 Performance: IBM Tivoli OMEGAMONA Deep Blue View of DB2 Performance: IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS*, SG24-7224, and the second edition of *DB2 9 for z/OS: Using the Utilities Suite*, SG24-6289.

Hyun Baek (also called Jimbo) is a DB2 for z/OS specialist and Information Management Tools technical sales specialist in IBM Korea. He joined IBM Korea Software Group 6 years ago and is an experienced DB2 for z/OS system programmer for the telecommunication industry. He also has experience with DB2 for z/OS performance benchmarking against other RDBMS products. He specializes in DB2 system performance tuning for data sharing, IBM CICS® and IBM DRDA® environments, and has worked on DB2 for z/OS data warehouse and business intelligence. Hyun coauthored *Enhancing SAP by Using DB2 9 for z/OS*, SG24-7239.

Rick Butler is a DBA Manager at BMO Financial Group, based in Toronto, Canada. He has 25 years of experience working with DB2 databases and mainframe banking applications. Prior to this, he worked in London, as a designer/programmer on a foreign exchange application. He also has expertise in DRDA. He holds a Bachelor of Science degree in Operations Research from Université de Montréal. Rick is coauthor of *The Business Value of DB2 UDB for z/OS*, SG24-6763 and *Securing DB2 and Implementing MLS on z/OS*, SG24-6480.

Ernie Mancill is a Senior Certified Executive IT Specialist with IBM Software Group. Ernie has 34 years of experience in IT with 19 years of experience with DB2 for z/OS as a Systems Programmer. He joined IBM ten years ago and is currently a member of the IBM SWG DB2 Database Tools technical sales team where he specializes in IBM Information Management Data Governance solutions. His areas of expertise include auditing, encryption, and other data governance solutions on DB2 for z/OS from IBM. Ernie co-authored *A Deep Blue View of DB2 Performance: IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS*, SG24-7224 and *Securing and Auditing Data on DB2 for z/OS*, SG24-7720.

The authors with Roger Miller in SVL. From left to right: Rick, Marcelo, Paolo, Ernie, Roger, and Hyun (photo courtesy of Megan Bock)

Thanks to the following people for their contributions to this project:

Rich Conway, Bob Haimowitz, Emma Jacobs
International Technical Support Organization

Madhavi Amirneni, Ron Ben-Natan, Jay Bruce, Gayathiri Chandran, Steve Chen, Al Cooley, Peter Dwyer, Xiaohong Fu, Lenisha Gandhi, Bala Iyer, Keith Howell, Irene Liu, Peter Mandel, Claire McFeely, Roger Miller, Jerry Mukai, Cristi Nesbitt Ullmann, Matthias Nicola, Jim Pickel, Cynthia Saracco, Ken Taylor, Derek Tempongko, David Zhang
IBM Silicon Valley Lab

George Markouizos and Mark Nelson
IBM RACF® Development, IBM Poughkeepsie

Norbert Heck and Norbert Jenninger
Boeblingen Lab, IBM Germany

Ernie Nachtigall
IBM Markham

Gregory Boyd
ATS - System z Crypto and eBusiness Security
IBM Gaithersburg

Now you can become a published author, too!

Comments welcome

2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Part 1

Security for DB2 for z/OS

In this part, we provide a discussion of the basic elements related to security for DB2 for z/OS.

This discussion starts with a review of general security concerns and concepts, with some specific application to the database environment. In today's business environment, there are a plethora of regulatory initiatives, many of which have specific implications for the DB2 environment. We review some of the more commonly encountered standards.

Next, we look briefly at the history of security implementations with DB2 on the mainframe environment.

This part contains the following chapters:

- Chapter 1, "Security regulations" on page 3
- Chapter 2, "Introduction to security for DB2 for z/OS" on page 19

1 Security regulations

This chapter discusses the security requirements for setting up secure DB2 for z/OS environments. We discuss the impact of data breaches and the various regulatory compliance initiatives, some of which are of specific concern to the DB2 10 for z/OS customer.

This chapter contains the following topics:

- The cost of a data breach
- Regulatory compliance initiatives

1.1 The cost of a data breach

For many organizations, data constitutes their most important and prized asset. Information can describe an institution's relationship with its customers, competitive or proprietary processes, trading relationships with partners, and tactical and strategic positioning against their competitors. Because of the close interaction with an organization's business, and the data that describes that business, it becomes a paramount requirement to protect that data. It is for all these reasons and more that when data is lost or stolen, real and significant damage can occur.

There have been a number of recent data breach incidents whose short and long term ramifications have been chronicled. Developing an understanding of these ramifications, and the potential costs associated with them, can help a company build a true picture of the risk and financial impact of an inadvertent or deliberate data breach. This can be used to assess the cost benefit that comes with implementation of additional security infrastructure, such as encryption and auditing solutions.

1.1.1 Actions after a data breach

One characteristic of data breach events and the subsequent application of state statutes, for the majority of states, has dictated that after a data breach occurs, the impacted individuals should be notified whenever their personal data has been affected. This can include personal data that has been stolen, lost, or compromised in some way.

It is this requirement for notification, mitigation of the associated follow-on consequences, and application of remedies that directly address the original source of the data breach that compose what is termed *direct costs*:

- Forensic analysis and internal investigation of the theft

  After the initial discovery of the incident is the effort to quantify the extent of the breach, understand the avenue used to execute the breach, and the institution of a short term remedy to close the source of the breach and avoid any additional loss of data. Among the important questions that need quick response is to quantify the extent of the exposure, the number of exposed individuals, the nature of information stolen, and to identify the specific population of exposed individuals.

  Note: While the requirements for breach notification can vary from state to state, many organizations might not be required to perform customer notification if the data breach involves encrypted data.

- Notification campaign, emails, phone calls, letters, and so on

  After the number and identities of the affected customer base has been discovered, each individual must be contacted to inform them of the specific nature of the data breach. This notification might need to include multiple means of contact, and in the absence of accurate forensic information, the community targeted in the notification campaign might be much larger than that directly impacted by the breach.

- Call center costs due to increased volume of customer traffic

  With the public acknowledgement of any data breach, and as the result of a successful notification campaign, there is a marked increase in the number of customer interactions requiring the intervention of the call center. This activity can tax the existing call center infrastructure and interfere with their ongoing support of normal customer services.

Temporary increases in call center capabilities need to be provided to manage this increased activity.

- Legal costs for defence and investigation

  To verify proper adherence to regulations and to help protect the organizations legal standing, external communications and response are subject to legal review.

- Internal investigations resulting in mitigation

  Along with the external response to contact and disclose the breach, inward inspection results in changes in procedures, implementation of new technology, and additional personnel to repair and better protect customer information from future breaches. Instead of a measured and planned implementation of these remedies, there is often a rush to judgement that results in a poorly planned and executed implementation. This could negatively impact other IT or line of business processes and result in increased use of IT and computing resources.

- Triage to salvage customer and investor relations

  There is increased scrutiny on the post-breach activities of an impacted organization by the customer base and the investor community. As the details of the data breach become more known and ultimate costs to an enterprise both factual and speculative become known, increased pressure is placed on the business to mitigate the concerns of the existing customer base.

- Fees and penalties

  Depending on the nature of the data breach, and in the presence of regulatory statutes or industry regulations, there could be significant costs levied in the way of fines and penalties. In some instances, there could also be legal ramification that could result in an organization's executives being subject to incarceration.

A data breach can have significant financial impact. However, much more damage can occur from the second class of damage, which we categorize as *indirect costs*. This category includes impact to a company's standing in the marketplace, and while the implications might not be readily apparent, the impact of a data breach can cause real and irrevocable damage.

- Loss of employee productivity

  Short term efforts to conduct forensics and implement triage to protect against future breaches require redirection of personnel. Deliverables and strategic initiatives can be delayed by this diversion of talent.

- Erosion of customer base due to loss of confidence

  A significant number of the affected customer population is at risk to terminate their relationship with any organization that allows personal information to be stolen. This erosion of confidence can expand by word of mouth to their network of acquaintances.

- Reticence for new customers to establish relationships

  Along with a loss of existing customers, any public disclosure of a data breach sends a message of a lack of institutional control and an apparent cavalier attitude of data custodianship to potential customers.

- Reduced shareholder confidence and value

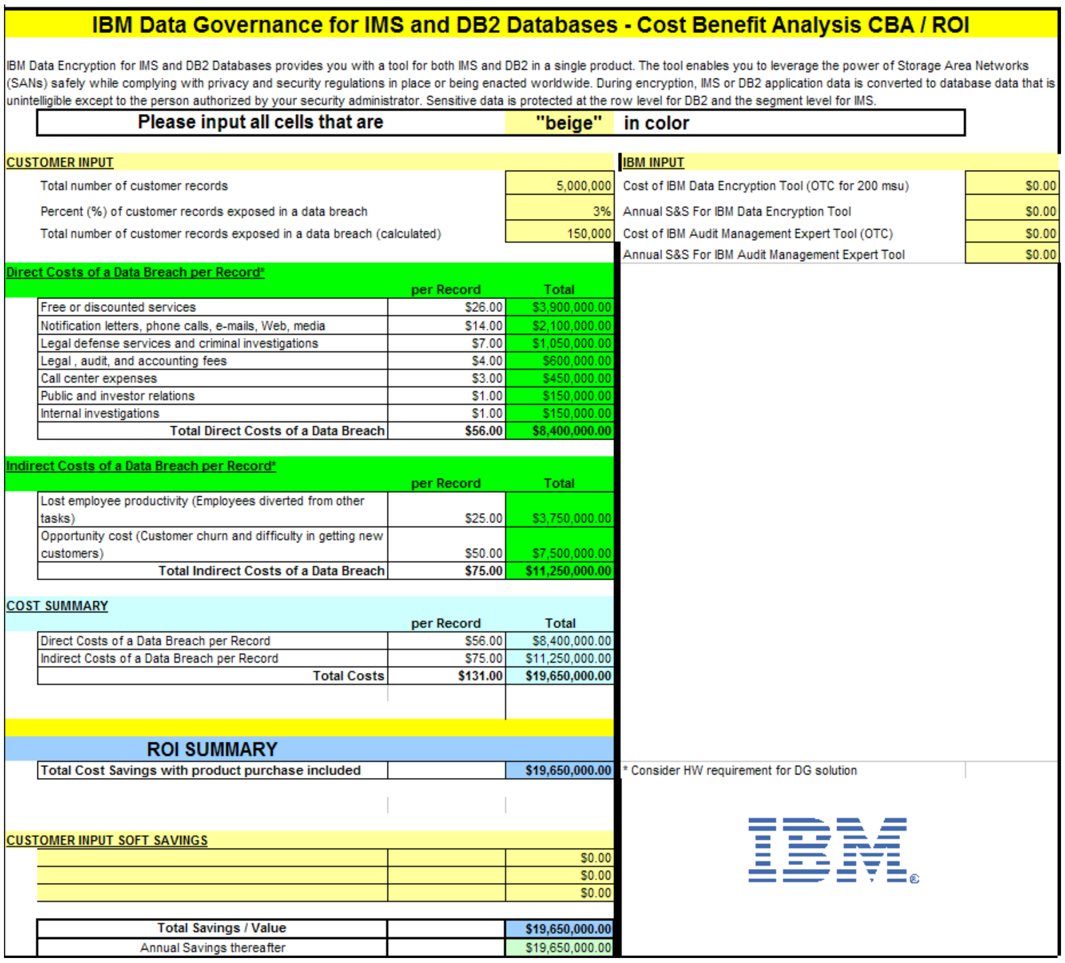
  As the financial ramifications of a data breach, and the necessary monetary investment to address and rectify the situation become known, there is a direct impact on the profit and loss position of the company. This can have ramifications that contribute to a perceived value of a company and the value of its stock.

- Decreased competitive standing

  Given the competitive landscape that most companies operate in, there is some significant damage to the brand name of any organization that suffers a public data breach. This presents some meaningful and compelling collateral that is used against a company by their competition. Brand image damage and erosion of the customer base might represent the most expensive downstream effect from a data breach.

1.1.2 ROI calculation

There can be severe financial repercussions that arise from a data breach. However, there is a corresponding cost associated with implementing protections, such as encryption or auditing database activity. While sometimes expensive in relation to the existing deployed technology, it is certainly cheaper to analyze the cost benefit and resultant justification before a breach occurs. To assist customers in the quantification of the cost of a typical data breach, IBM has created the IBM Data Governance ROI calculator shown in Figure 1-1.



IBM Data Governance for IMS and DB2 Databases - Cost Benefit Analysis CBA / ROI

IBM Data Encryption for IMS and DB2 Databases provides you with a tool for both IMS and DB2 in a single product. The tool enables you to leverage the power of Storage Area Networks (SANs) safely while complying with privacy and security regulations in place or being enacted worldwide. During encryption, IMS or DB2 application data is converted to database data that is unintelligible except to the person authorized by your security administrator. Sensitive data is protected at the row level for DB2 and the segment level for IMS.

Please input all cells that are "beige" in color

CUSTOMER INPUT		IBM INPUT	
Total number of customer records	5,000,000	Cost of IBM Data Encryption Tool (OTC for 200 msu)	$0.00
Percent (%) of customer records exposed in a data breach	3%	Annual S&S For IBM Data Encryption Tool	$0.00
Total number of customer records exposed in a data breach (calculated)	150,000	Cost of IBM Audit Management Expert Tool (OTC)	$0.00
		Annual S&S For IBM Audit Management Expert Tool	$0.00

Direct Costs of a Data Breach per Record*	per Record	Total
Free or discounted services	$26.00	$3,900,000.00
Notification letters, phone calls, e-mails, Web, media	$14.00	$2,100,000.00
Legal defense services and criminal investigations	$7.00	$1,050,000.00
Legal , audit, and accounting fees	$4.00	$600,000.00
Call center expenses	$3.00	$450,000.00
Public and investor relations	$1.00	$150,000.00
Internal investigations	$1.00	$150,000.00
Total Direct Costs of a Data Breach	**$56.00**	**$8,400,000.00**

Indirect Costs of a Data Breach per Record*	per Record	Total
Lost employee productivity (Employees diverted from other tasks)	$25.00	$3,750,000.00
Opportunity cost (Customer churn and difficulty in getting new customers)	$50.00	$7,500,000.00
Total Indirect Costs of a Data Breach	**$75.00**	**$11,250,000.00**

COST SUMMARY	per Record	Total
Direct Costs of a Data Breach per Record	$56.00	$8,400,000.00
Indirect Costs of a Data Breach per Record	$75.00	$11,250,000.00
Total Costs	**$131.00**	**$19,650,000.00**

ROI SUMMARY

Total Cost Savings with product purchase included		**$19,650,000.00**

* Consider HW requirement for DG solution

CUSTOMER INPUT SOFT SAVINGS		
		$0.00
		$0.00
		$0.00
Total Savings / Value		**$19,650,000.00**
Annual Savings thereafter		$19,650,000.00

Figure 1-1 Data Governance Calculator - Encryption Tool for DB2 and IBM IMS™ Databases

Using documented industry information, the calculator breaks down the potential impact of a data breach into a number of categories, representing both direct and indirect cost categories, which were discussed in the previous section. The information provided by the customer is a number of customers or records that describe information of interest to a potential attacker.

In addition, an estimate is provided that describes the percentage of records that become exposed in the event of an exposure. The result is a breakdown of the line item details of the cost of the data breach. Armed with this information, you can clearly substantiate the cost of security remediation against the potential financial damages that come with a data breach.

Databases house a wide variety of data, all important to individual organizations. This information can range from financial data, private customer information, intellectual property, sensitive employee data, and more. In many cases, this data forms the most valuable asset a company has, and there is nothing more damaging (from an information security perspective) than a compromise of this asset.

DB2 for z/OS provides a rich set of features to help ensure robust data protection. However, these features are not always used and are not always used correctly. Security issues are often a matter of misconfiguration, and the fact that DB2 for z/OS is built on a robust security model does not mean that it is being used or that it is being used correctly. While DB2 for z/OS been around for a long time, the following trends have been emerging in the past few years:

- E-commerce and e-business

  E-commerce and e-business have brought a paradigm shift to the way in which organizations interact with their customers and, in a fundamental way, the lives that we live each day. We buy from online retailers, we pay our utility bills using online banking sites, and more. Businesses have optimized their supply chains and use Customer Relationship Management (CRM) software to manage relationships with their clients. Systems have become much more interconnected, and have moved much closer to the external users.

  Network configurations can use firewalls to secure our networks, and while databases are not directly connected to the Internet, we can demonstrate that databases are far more exposed than they used to be. Ten years ago, the database was accessed by applications that were only available to internal employees, through well protected OLTP applications such as CICS and IMS. Now it is, indirectly through the application, accessed by anyone who has access to the Web, which is pretty much everyone in the world. E-business has also had a much bigger impact on security (or the lack of it). E-business has brought the potential for adding many indirect users of the database.

- New features and database functionality

  New features of DB2 for z/OS, such as increased XML support in the database, and better usage of the Java environment, has opened up new ways in which the database is used and innovative ways in which it is configured. Using these new database features to open the enterprise to improve processes, enhance the customer experience, and streamline business can now be done quickly, and in some cases, without too much analysis of security implications. Databases are deployed in many places (physically and logically) and without thinking about the usage of the protective layers provided by DB2 for z/OS, represent significant risks. These technologies include web services within the database, XML handling within the database, tight integration with application servers, and the ability to run any application logic directly within the database (to the extent of having an embedded Java virtual machine inside the database). This functionality is great for developers and for increasing productivity, but it creates a security nightmare. More functionality means more (actually, many more) bugs that can be exploited by hackers and many of the leading vendor databases have been plagued with bug-related vulnerabilities. Even if new functions have no vulnerabilities, these features are usually risky because they open the database to more types of attacks. They increase not only the developer's productivity but also the hacker's productivity.

- Increased awareness among the hacker community.

  Hackers are always looking for new targets for their attacks and new methods they can use. In the same way that you realize that databases hold the crown jewels, so do the hackers. Furthermore, after mastering attacks on networks and operating systems, hackers have turned to applications and databases as a new breeding ground. This is visible in hacker forums.

- Widespread regulations that pertain to IT and to security.

  Bad accounting practices, fraud, and various corporate scandal and crimes have prompted regulators to define and enforce new regulations that have a direct impact on IT auditing. Because financial, personal, and sensitive data is stored within databases, these requirements usually imply database auditing requirements. Because regulations such as Sarbanes-Oxley, HIPAA, and GLBA have financial and criminal penalties associated with noncompliance, interest in database security and auditing have suddenly come to the forefront.

1.2 Regulatory compliance

While there are many security products and methodologies, technology is not enough. What is also required is a willingness to address the problems and invest in security solutions that will guarantee the security and privacy of information. This willingness does not always exist because of limited budgets. Some people point to the fact that security does not always display a clear return on investment (ROI), but neither does an alarm system you may install at home or insurance you pay every year.

Leading companies understand that in the same way that people continue to protect and insure houses and cars, they must continually invest in protecting valuable information. One incident that involves theft or destruction of proprietary information can easily pay for a 10-year investment, and serious incidents can cripple a company for life. For those companies that have not come to this realization, regulators have created a large (and growing) set of regulations and frameworks aimed at enforcing protection of information, privacy, and transparency of information.

These regulations have come into existence in the past couple of years (and will continue to do so) have been prompted by some significant damages made to companies, and more important, to the public.

Some of these regulations, such as HIPAA for health care and GLBA for financial services, are specific to certain market segments. Others are for a certain class of companies, such as Sarbanes-Oxley for public companies and California Senate Bill 1368 (SB 1368) for companies that maintain personal information regarding residents of California. In all cases, these regulations include stringent requirements dealing with information security and privacy, and all of them implement punitive consequences if compliance is not maintained.

One example we can review is the Sarbanes-Oxley Act (SOX) of 2002. Questionable accounting practices and poor management in companies such as Enron and Worldcom shattered investor confidence and caused Congress to pass the Sarbanes-Oxley Act of 2002 (SOX). While some companies are reacting to SOX by addressing minimum requirements that had to be made by the end of 2004, other companies are also addressing requirements that will take effect at a later date. Among these are requirements for real-time disclosure of any event that may affect performance, as well as security and privacy issues.

While SOX compliance is primarily the responsibility of the CEO and CFO, CIOs have a key role in implementing technology strategies that can support real and implied integrity, security, credibility, and transparency requirements that SOX has defined, both for financial systems and for other systems that manage data that is critical to company performance, including ERP, CRM, SCM, and so on. Because all of these systems employ relational databases, where the data is actually stored eventually, these projects include database security and auditing implementations. You need to understand what these regulations are all about (and how to deal with them).

1.2.1 Health Insurance Portability and Accountability Act of 1996

The Health Insurance Portability and Accountability (HIPAA), also known as Public Law 104-191 or the Kennedy-Kassenbaum Bill, is an act passed by the U.S. Congress and signed into effect on August 21, 1996. HIPAA's general objectives are as follows:

- Guarantee health insurance coverage of employees.
- Reduce health care fraud and abuse.
- Implement administrative simplification to augment effectiveness and efficiency of the health care system.
- Protect the health information of individuals against access without consent or authorization.

The act requires that U.S.-based health care companies be HIPAA-compliant by October 2003. If you live in the United States, you will have noticed that around that time (and even today) every time you go to a new provider you are asked to sign some HIPAA document. This is mandated by section 164.520-Notice of privacy practices for protected health information. This section states that providers and other entities must provide individuals with a notice of privacy practices, that the notice must be in “plain language,” and that it must include clauses such as the following:

- Information about uses and disclosures of protected health information
- An explanation of privacy rights
- How to file complaints

HIPAA addresses problems in the way health care companies, providers, and carriers do business among them and specifically in the way that data is used and stored, and the way transactions are performed. HIPAA tries to address many problems that are prevalent in the U.S. health care system and make the system frustrating at times. These problems include the difficulty in sharing information among providers, incorrect information in outdated repositories, the inability to share data because of misaligned formats and representations, and perhaps most important, leakage of patient information.

HIPAA is a requirement for any organization that deals with patient information in the United States. It includes all health care providers and other entities that are part of the health care service chain; these are collectively called Covered Entities (CEs) by HIPAA. Providers include all hospitals, doctors, clinics, social services programs, and even schools (because they provide immunizations). Other entities include Medicare, Medicaid, health insurance companies, life insurance companies, and even all employers to some degree.

Most of HIPAA addresses policies and procedures, but a sizable chunk deals with technology. HIPAA contains five main sections that address the following areas:

- Health care access, portability, and renewability
- Administration simplification, fraud and abuse prevention, and medical liability reform
- Group-health requirements

- Revenue offsets
- Tax-related provisions

The area in which IT comes up the most is that of administration simplification. More specifically, there are four main areas that touch on technology:

- Privacy of patient information. HIPAA mandates that medical records and patient information should be protected. Furthermore, HIPAA sets penalties for information leakage which be up to $250,000 per incident and up to 10 years of imprisonment of the executive in charge.
- Verifiable security policies. HIPAA mandates that health care organizations have a clear, verifiable, and auditable security policy. It also mandates that organizations perform privacy risk assessments and train employees in privacy procedures.
- Patient's access to information HIPAA requires that patients can always access their private information in a standard format and that this information be readily available (upon the patient's request) to other doctors, providers, and so on. This is the *portability* in HIPAA.
- Standardized information exchange. HIPAA mandates that information related to insurance should be exchanged in a standard, predefined way.

It is interesting to note that HIPAA addresses and mandates two separate issues that are somewhat opposite. The first two requirements deal with protecting information and ensuring privacy. The latter two deal with the need to be able to get this information to authorized entities quickly, easily, and with no information-related barriers. The coupling of these seemingly opposed issues is intentional. HIPAA recognizes the fact that by mandating that patient records be sent over networks, there is a risk that patient privacy could be compromised. To address this risk, the Department of Health and Human Services developed a standard set of security and privacy regulations with which CEs must comply. All of the sections mentioned may be important to you as the database owner. The main sections you need to understand and deal with are those that specifically mention and deal with privacy of patient information and those that discuss implementing an auditable security policy.

The security requirements outlined in HIPAA require the following items:

- Management involvement in the development and implementation of HIPAA-compliant security policies and procedures
- Periodic review of these policies and procedures
- Training on policies and procedures for all employees who come in contact with private patient information
- Technical measures that are integrated into the organization's information systems

1.2.2 Gramm-Leach-Bliley Act of 1999

The Gramm-Leach-Bliley Act (GLBA) was enacted on November 12, 1999, approximately seven months after the merger between Citicorp and Travelers Group to form Citigroup. GLBA (sometimes also called the *Citigroup Relief Act*) allows financial holding companies like Citigroup to own banks, insurance companies, and securities firms. Before GLBA, operation of an insurance underwriter (Travelers) was not allowed for a bank holding company. To make matters even more complex, Travelers owned Salomon Smith Barney, and its bank-ineligible activities composed more than the allowed 25%.

When GLBA came along, it created a new definition of a Financial Holding Company (FHC) that allowed Citigroup to exist. Luckily, GLBA is not one-sided. It did allow for the creation of mega-financial companies, but it went on to define limitations and requirements on these FHCs. Some of these requirements are based on capitalization (that is, the need to remain well-capitalized and maintain a high rating). Other limitations are in the area of privacy.

One of the main reasons for creating mega-financial companies is to leverage a knowledge base and be able to do cross-selling within the FHC. If an insurance company just merged with a large bank, the insurance company can try to market its products to all customers of the bank. However, there is the risk that the collective set of data that exists within the FHC about individuals can be large, in which case any leakage can be more damaging to the individual.

To combat extreme misuse of such cross-selling and the risks to privacy, Congress adopted Title V of GLBA, which defines various requirements designed to protect the privacy of customers of financial institutions. This is the main relevance GLBA has in the context of database security and auditing. Title V includes both the Financial Privacy Rule and the Safeguard Rule. The Financial Privacy Rule discusses operations and practices, while the Safeguard Rule has a more technical interpretation and includes requirements for the following activities:

- Ensure the security and privacy of customer information.
- Protect against threats to the security and integrity of customer information.
- Protect against unauthorized access or usage of this information that could result in harm or inconvenience to the customer.

1.2.3 Sarbanes-Oxley Act

The Sarbanes-Oxley (SOX) Act of 2002 was passed by the U.S. Senate and the U.S. House of Representatives with large majorities and signed into law on July 30, 2002. It is the U.S. government's answer to increasing concern and heightened awareness of corporate governance, conflicts of interest, and the lack of financial reporting transparency that seems to have plagued the U.S. corporate landscape and has caused significant damage to investors. SOX applies to any public company (including non-U.S. companies). Because of this wide definition, SOX is perhaps the most visible regulation, and therefore most companies have (and will have) significant projects and money allocated to becoming compliant with SOX.

SOX addresses many areas that have in the past, and may in the future, affect the accuracy and transparency of financial reporting. Many of these provisions have nothing to do with databases or other technical issues. Many of the provisions deal with board member and executive management issues so that, for example, CEOs will not be able to work with a Chairman of the Board (sometimes forming the majority of the compensation committee) to approve a bonus to the CEO and a new pool of options to the Chairman of the Board. At a high level, the topics that SOX regulations address include the following:

- Audit committee issues
- Audit committee expertise
- Enhanced review of periodic disclosures
- New oversight board for corporate governance
- Certification of financial statements
- Improper influence of conduct of audits
- Forfeiture of bonuses and profits (in some cases)
- Off-balance sheet transactions

- Pro-forma financial information
- Dealings with securities analysts

The most important topic in this bill relevant to our discussion is the certification of financial statements: CEOs and CFOs are required to personally sign and certify the correctness of financial reports. They need to attest that to their knowledge the filed reports do not contain any untrue statement or omission and that they represent the true financial condition of the company. They are personally responsible for the report and can even go to jail if a few years down the line the company needs to restate financial reports (as has been done often in the past few years) as a result of improper information presented in financial reports, especially if they cannot prove that they took enough steps to try to ensure that the information was correct.

SOX is a detailed document, and you do not really need to read the whole of it. The most important section (and the one most IT people focus on) is Section 404, which requires management to report on the effectiveness of the company's internal control over financial reporting. This section requires management's development and monitoring of procedures and controls for making assertions about the adequacy of internal controls over financial reporting. Furthermore, it is management's responsibility and cannot be delegated or abdicated, so they also need to understand what is being audited, monitored, and how control is enforced (that is, they cannot just be told that everything is correct). It goes even further: Management has to document and evaluate the design and operation of, and report on the effectiveness of, its internal controls. Management has to document the framework used, assess its effectiveness, publish any flaws and weaknesses, and do all of this within the annual report published to investors. This boils down to the need for visibility, transparency, and segregation of duties

1.2.4 California Senate Bill 1386

In September 2002, the Governor of California signed Senate Bill 1386 into effect. Among other things, SB 1386 mandates that:

"operative July 1, 2003, . . . a state agency, or a person or business that conducts business in California, that owns or licenses computerized data that includes personal information, as defined, to disclose in specified ways, any breach of the security of the data, as defined, to any resident of California whose unencrypted personal information was, or is reasonably believed to have been, acquired by an unauthorized person. . . . For purposes of this section, "breach of the security of the system" means unauthorized acquisition of computerized data that compromises the security, confidentiality, or integrity of personal information maintained by the agency."[1]

In effect this means that any business that maintains personal information of a resident of California must have the appropriate provisions and capabilities to know when this information may have been accessed by an unauthorized person. This bill adds to a long line of bills that focus on privacy, but stresses not just the need for privacy but also the need for effective controls that will allow you to know when access control has been compromised and data has been accessed in an unauthorized manner.

[1] Source: http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html

1.2.5 Payment Card Industry Data Security Standard

The use of payment cards as a form of currency in exchange for goods and services is the cornerstone of the infrastructure supporting economic growth around the world. In the United States alone, the estimated 641 million credit cards in circulation account for about $1.5 trillion in consumer spending each year.

However, high-profile data breaches have exposed the vulnerability of payment card processors, point-of-sale vendors, and financial institutions that are not properly securing confidential customer information. Facing increasing risk and financial losses resulting from the misappropriation and misuse of customer information, the payment card industry has taken the initiative. The Payment Card Industry Data Security Standard (PCI DSS) represents the payment card industry's response to these breaches. Merchants and retailers who cannot protect consumer payment card information will be held accountable.

The PCI DSS Council was formed by the major payment card brands (American Express, Discover Financial Services, JCB International, MasterCard Worldwide, and Visa Inc.) to provide a forum in which all stakeholders can provide input into the ongoing development, enhancement, and dissemination of the PCI DSS, PIN Entry Device (PED) Security Requirements, and the Payment Application Data Security Standard (PA-DSS). For more information about he PCI DSS, visit the following website:

https://www.pcisecuritystandards.org/

You can download the specifications document (a set of comprehensive requirements for enhancing payment account data security) from the website after signing the licence agreement.

The PCI DSS is a multifaceted set of regulations that defines requirements for implementing security management policies, procedures, network architecture, software design and other critical protective measures. With the goal of improving the security of electronic payments, the PCI DSS represents a unified industry standard for protecting cardholder data that is stored, transmitted, or processed.

The standard includes 12 requirements across six categories, concentrating on data authentication, access control, audits, and data encryption. To comply, companies that handle payment card information are required to establish stringent security policies, processes, and procedures.

The standard covers a range of issues, such as maintaining a secure network, protecting cardholder information, managing risk, implementing control measures, and monitoring test networks. See Table 1-1.

Table 1-1 PCI DSS requirements

Requirement	Description
A - Build and maintain a secure network.	
1	Install and maintain a firewall configuration to protect cardholder data.
2	Do not use vendor defaults for system passwords and other security parameters. Use configuration standards (NIST/SANS/CIS).
B - Protect cardholder data.	
3	Protect stored cardholder data.
4	Encrypt transmission of cardholder data across open, public networks.

Requirement	Description
C - Maintain a vulnerability management program.	
5	Use and regularly update antivirus software.
6	Develop and maintain secure systems and applications.
D - Implement strong access control measures.	
7	Restrict access to cardholder data by business need-to-know.
8	Assign a unique ID to each person with computer access.
9	Restrict physical access to cardholder data.
E - Regularly monitor and test networks.	
10	Track and monitor all access to network resources and cardholder data.
11	Regularly test security systems and processes.
F - Maintain an information security policy.	
12	Maintain a policy that addresses information security.

In this section, we examine the six categories of PCI DSS requirements from the point of view of the solutions of the IBM System z® and security-provided functions.

Objective A: Build and maintain a secure network

Objective A has two components:

- Install and maintain a firewall configuration to protect cardholder data.
- Do not use vendor defaults for system passwords and other security parameters. Use configuration standards (NIST/SANS/CIS).

IBM countermeasures

System z contains many elements that introduce controls and protections that can be viewed as the equivalent of firewall technology. Keep in mind also the tremendous diversity of workload that can be supported by System z, and reap the benefit of this technology.The forebears of System z and the z/OS operating system introduced a fundamental architecture that enforces resource isolation and limits memory access across different application processes through logical partitioning. The z/OS Network Policy Agent and z/OS System Health Checker, along with IBM Tivoli® zSecure Admin, can provide mechanisms to verify that critical parameters are changed from the well-known vendor-supplied default values. All processes that run on System z are subject to control by the IBM Resource Access Control Facility (RACF). RACF administration best practices can be enhanced with the use of the Tivoli zSecure Audit Admin product.

The security features that are inherent with System z and z/OS have been well chronicled, and are viewed as being highly secure. System z can be considered an excellent place to build a layered defense for a web-facing demilitarized zone (DMZ), especially with the use of System z IFL (Integrated Facility for Linux). System z has been granted some of the highest levels of security certification, including Common Criteria. The System z implementation of logical partitioning (LPAR) has obtained an Evaluation Assurance Rating of 5 (EAL5)[2], the z/OS Operating System provides a EAL rating of 4+, and IBM z/VM® has an EAL5 rating as well.

[2] The Evaluation Assurance Level (EAL1 through EAL7) of an IT product or system is a numerical rating assigned at the completion of a Common Criteria security evaluation.

Objective B: Protect cardholder data

Objective B has two components:

- Protect cardholder data.
  - Key management and key rotation
  - Certificate management
- Encrypt transmission of cardholder data across open, public networks.
  - Certificate management
  - Dumps

IBM countermeasures

All of the elements needed to store and process cardholder data effectively, along with the robust security to protect it, are available within the hardware elements of System z. The z/OS Communications Server, a component of the z/OS operating system, provides System z hosted applications the ability to communicate across the TCP/IP stack using IBM HiperSockets™, which provide high speed capability without exposure through an open or public network. The z/OS Communications Server also provides intrusion detection and protection. Intrusion Detection Services (IDS) evaluates the network stack for attacks that would undermine the integrity of its operation.

The fourth requirement addresses encryption. It states that sensitive data must be encrypted while in storage (data at rest) and when cardholder data is transmitted across open public networks. System z provides for strong encryption support. The implementations are dependent on what hardware elements are available on the System z and what type of encryption is needed, generally chosen by business requirements.

To protect cardholder data with encryption, some implementations rely on multiple platforms. These implementations have poor coordination and can be viewed as having little consistency. When having to coordinate encryption support and associated activities across multiple platforms, this becomes a point of vulnerability. System z has capabilities to manage encryption across heterogeneous environments.

As mentioned earlier, there is a long-standing heritage of features within the z/OS operating system that can contribute to securing data stored on System z. (Some of these elements include Storage Protection Keys, Cross-Memory Services, enforced workload isolation, z/OS Workload Manager, RACF, and z/OS Communications Server.) System z contains memory control mechanisms. In particular with z/VM, as memory is swapped out for use by other applications (part of multi-processing), the memory gets erased. Other platforms that claim to support virtualization through the use of virtual machines cannot make this claim.

DB2 offers a regulatory compliance suite with tools to encrypt, test, audit, and safely archive data for long-term retention. These tools include IBM InfoSphere™ Guardium S-TAP for DB2 for z/OS to provide a deep level of auditing capabilities for DB2, and IBM Encryption Tool for IMS and DB2 Databases to help implement enterprise class encryption for data at rest while using the latest in System z encryption hardware technology.

The following information pertains to the over-the-network protection by the hardware or within z/OS:

- DB2 9 features the implementation of SSL, AT-TLS, or IPSEC encryption for sending and receiving data.
- z/OS supports SSL, TLS, AT-TLS, IPSec, OpenSSH, and Open-PGP, plus multiple symmetric and asymmetric encryption methods, including TDES and AES.
- z/OS Communications Server provides z/OS Intrusion Detection Services to complement network-based IDS. It can detect known and unknown attacks, and can detect problems in real time, providing another layer of network defense.

- RACF provides the facilities to manage access and disallow untrusted networks and hosts.
- Routing IPSec activity to a specialty zIIP processor, supported in z/OS 1.8, reduces the cost of processing in the same way System z redirects Java execution to one of its zAAP specialty processors. This improves the price/performance of end-to-end encryption.
- FTP, which is cited as a risky protocol in PCI-DSS requirements, can be protected on System z with IPSec, AT-TLS, or SSL

The Integrated Cryptographic Service Facility (ICSF) is a component of z/OS. CP Assist for Cryptographic Function (CPACF) is a feature available on IBM System z9® and IBM System z10®, which, while a non-chargeable option, requires that the hardware Customer Engineer (CE) enable its use. When enabled, CPACF adds cryptographic machine instruction support to every general purpose processor, making routine use of cryptography more transparent. For clear key encryption requests, much better performance characteristics can be achieved.

In addition to the CPACF facility, which runs on the general purpose processors, there is an additional hardware element, the Cryptographic Express2 Coprocessor (CEX2C) feature, that can be added for a separate cost. This feature is used to support secure key encryption. All secure key cryptographic work is performed within the hardware boundaries of the CEX2C feature, which provides a completely isolated environment, and at no point in time is any data or cryptographic key exposed to operating system storage. It is also referred to as tamper resistant. In the event of someone gaining access to the CEX2C hardware, upon attempting to remove the element from the hardware cabinet, the registers containing the master key values are zeroized[3], thereby protecting the key from inadvertent or intentional exposure or theft.

In conjunction with the CEX2C, you can also perform third-party digital certificate hosting using PKI Services element of z/OS. This element allows z/OS to enable enterprises to become their own certificate authority, which not only reduces the cost associated with relying on an outside service to provide certificates, but also creates a more secure implementation, as it eliminates a trip out through the public network, which introduces an additional security exposure.

Objective C: Maintain a vulnerability management program

Objective C has two components:

- Use and regularly update antivirus software or programs.
- Develop and maintain secure systems and applications:
  - Latest vendor supplied patches (getting them)
  - Separate development and test environments
  - Review of custom code
  - Change management procedures

IBM countermeasures

Satisfying the fifth requirement to maintain antivirus protection is somewhat less important or difficult on System z than on other platforms because System z performs antivirus processes as part of its operations. The PCI-DSS requirements note that the mainframe, by its nature, is not vulnerable to viruses to the extent that Intel platforms are, so it is commonly viewed that for PCI, applications and operating system environments on System z are not subject to exposure to a virus.

Requirement six, to secure systems and applications, is covered by SMP/E, the z/OS software maintenance and installation tool, LPARs (EAL5), Storage Protection Keys, and System z isolation of sensitive executables as part of its processing routine.

[3] The stored data is erased or overwritten.

IBM Optim™ Data Growth provides the ability to archive relationally intact pieces of inactive data from a customer's operational data store. Optim Test Data Management provides a capability to build right-sized representative test data copies with relational integrity, and in conjunction with Optim Data Privacy, provides the capability to mask all of the PCI-DSS sensitive data elements (the Primary Account Number (PAN) and associated elements).

On a mainframe, in a well-defined security environment, security operations (RACF administration) are separated from systems administration. This separation of roles and responsibilities is as much a part of business prudence as a technology choice. It is like having the person who signs the checks also has the ability to print them. In most implementations, systems administrators have low levels of authority to effect changes to the security environment, and the security personnel have limited system administration privileges.

Objective D: Implement strong access control measures

Objective D has three components:

- Restrict access to cardholder data on a business need to know.
- Assign a unique ID to each person with computer access.
  - Two-factor authentication
  - Root/SCHED
- Restrict physical access to cardholder data.

IBM countermeasures

Under System z, each piece of work is associated with an identity, usually a RACF-assigned identifier, and subsequent access to resources and facilities are based on permissions granted to that identifier by the RACF administrator. Most facilities that provide points of entry or access to System z resources require an authentication exchange of credentials, which includes a password authentication process. There is no way for an interloper with nefarious intent to bypass these controls.

In the System z environment, removable media (such as tape) are protected by access controls, and include tape label checking and verification. There is also a mechanism providing for the processing of *foreign tapes* (tapes created on non-System z systems or external System z processors). This mechanism is typically tightly controlled by the RACF administrator and must be implicitly invoked by RACF-authenticated users granted access to this facility.

RACF and Tivoli zSecure provide the protection and the means to monitor and audit access to protected resources by both standard and privileged users. This meets Requirement 7 and 9, which mandates restriction of access to those with a business need. Part of an implementation that demonstrates this requirement would be to limit the number of privileged users to the absolute minimum number needed to operate the System z environment.

Requirement 8 addresses the assignment of a unique ID to each person with computer access. RACF password management includes rules for password values (enhancing data security) and expiration of passwords. In addition, DB2 and IBM WebSphere® Application Server (host to many J2EE applications) can be configured to share a trusted context, making that environment more secure.

Objective E: Regularly monitor and test network

Objective E has two components:

- Track and monitor all access to network resources and cardholder data, which includes File-integrity monitoring.

- Regularly test security systems and processes, which also includes file-integrity monitoring.

IBM countermeasures

Monitoring and testing is best done from a point of security or the whole process is less than secure. IBM offers a plethora of monitoring tools such as RACF that can generate audit records for both successful and failed access attempts. System HealthChecker (built into z/OS), SMF auditing, Tivoli zSecure, Tivoli Consul Insight Manager, and InfoSphere Guardium S-TAP for DB2 for z/OS can interface with enterprise-wide auditing through the Tivoli platform to monitor and test the environment wherever it extends. Of course, the more that activity involving customer information is done on System z, the less complex the compliance with PCI-DSS.

Objective F: Maintain an information security policy

Objective F has one component, which is to maintain a policy that addresses information security for employees and contractors through a business continuity plan.

IBM countermeasures

System z and its use of RACF provides the foundation that allows the customer to create and enforce a security policy for information security served by z/OS and System z. In addition, this foundation is significantly enhanced by IM Tools for DB2 and IMS, Tivoli, Insight Suite, and z/OS Network Policy Agents. As platform decisions are made, particularly when choosing an infrastructure to host a new application workload, deploying these applications on System z will significantly reduce the security implications of an infrastructure choice.

It is equally important that you understand that there are going to be different security policies for different types of information and for different platforms. As enterprises succeed and grow through acquisition and customer base expansion, there will be new sources of data, new challenges in data security, and the desire to use existing security elements already deployed. Without a unified security policy, additional complexity can be introduced. IBM believes that the breadth of security elements imbedded within System z and z/OS provide an extremely robust and secure foundation, and with additional products to enhance the native operating system and sever based security, provide a world class environment for secure data hosting.

1.2.6 IBM Data Server Security

Securing data requires a holistic and layered approach that considers the broad range of threats. This is commonly referred to as *defense in depth*, and requires a *security by design* approach, where multiple layers of security work together to provide the three ultimate objectives of security, commonly known as the CIA triad: confidentiality, integrity, and availability.

IBM understands these data security threats, and designs security features directly into its DB2 and IBM Informix® families of data servers. Both data server families are designed with a wide range of security and auditing capabilities to help protect even the most critical data.

For best practices with IBM data servers, refer to the white paper *Best Practices IBM Data Server Security*, found at:

http://public.dhe.ibm.com/software/dw/dm/db2/bestpractices/DB2BP_Security_1010I.pdf

2

Introduction to security for DB2 for z/OS

In this chapter, we introduce security topics for DB2 for z/OS.

This chapter contains the following topics:

- ► DB2 and z/OS threat environment
- ► Application versus DBMS compliance controls
- ► Privileged user controls
- ► DB2 for z/OS from an evolutionary perspective

2.1 DB2 and z/OS threat environment

In general, customers have experienced far fewer security problems with System z machines than with any other DBMS hosting operating and hardware platform. It has been said many times that if z/OS was as popular as Windows and exposed to the Internet, it would have just as many security issues. The Internet is a hostile environment that poses special challenges for system designers, and the improvements to System z now encourage the hosting of Java applications and web-facing applications. The superior stability and security of z/OS and System z is real and has been demonstrated since the early 1970s. Ethical hacking experiments have confirmed that z/OS is highly resistant to malicious attacks. The unique technology provided by mainframes was designed and architected early in the IBM mainframe evolution to avoid the security and integrity problems that plague other platforms. z/OS interacts with System z hardware to provide control on the actions of applications and help ensure a high level of system integrity.

2.1.1 System z and DB2 for z/OS statement of integrity

IBM recognizes the strength of the platform and also realizes that it can be used in a wide variety of complex environments, each potentially bringing its own set of problems. IBM has great confidence in the protection controls supplied by System z hardware and the z/OS operating system software and consequently has produced a statement of integrity. This statement was first produced in 1973 for the forerunner of z/OS, which at the time was simply known as IBM MVS™. The current statement of integrity can be found at:

http://www.ibm.com/systems/z/os/zos/features/racf/zos_integrity_statement.html

The main aim of the statement is to demonstrate the degree to which the operating system and the hardware provide a solid reliable base for security controls. For example, it is important that the identity of a user is established and then is not capable of being changed by that user or any other user of the z/OS operating system. Thus, the responsibility for establishing user identities must be performed by the operating system or one of its components using code that runs in a privileged state.

2.1.2 Buffer overflow and storage protection on z/OS

Buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. This is a special case of violation of memory safety.

Buffer overflows can be triggered by inputs that are designed to execute code, or alter the way the program operates. This may result in erratic program behavior, including memory access errors, incorrect results, a crash, or a breach of system security. They are the basis of many software vulnerabilities and can be maliciously exploited. In some cases, the threat of the vulnerability is simply Denial of Service, while in other situations the remote execution of arbitrary code is possible.

In storage management, as implemented on System z and z/OS, for protection purposes, main storage is divided into blocks of 4096 bytes. A four-bit storage protection key is associated with each block. When data is accessed in a storage block, the storage block key is compared with a protection key. A specific protection key is assigned to each process or instruction. Problem state processes usually share protection key 8. Supervisor state processes and system processes often use different protection keys.

A buffer overflow into an adjacent 4 KB block can only occur if this block happens to have the same storage protection key assigned. A buffer overflow from a user program into a supervisor area is essentially impossible. This implementation of hardware key protection is only available on System z. It effectively blocks buffer overflow and unauthorized kernel access.

In addition to storage protection through a storage protection key, System z and z/OS provides other storage protection mechanisms, all of which block buffer overflows and unauthorized access to operating system elements. The full set of memory protection mechanisms used within z/OS include:

- Key-controlled protection
- Page protection
- Address space list-controlled protection
- Low address protection

2.1.3 Storage keys and the Authorized Program Facility

Instructions execute on System z using a construct known as the Program Status Word (PSW). The PSW includes the instruction address, condition code, and other information to control instruction sequencing and to determine the state of the CPU. If the non-zero storage key in the PSW does not match the key on the page and the fetch-protection bit is on, then access is denied regardless of the type of action. In addition to controlling what storage can be accessed, the PSW contains a bit known as the *state bit*. This bit is used to determine which set of instructions can be executed. When this bit is on the machine, it is said to be in the *problem program state* and is restricted in the instructions it may execute. However, when the bit is off, the machine is said to be in the *supervisor state* and may execute all instructions. Many of the instructions that require the supervisor state are designed for the use of the operating system. The general instructions (that is, those that are not privileged) are used to perform operations on data within a specific address space and execute against data that is private to the executing application or program. General instructions can be used in both in the supervisor state and in the problem program state.

There is one other rule regarding keys that is important in z/OS. If the key in the PSW is any value between 0 and 7 inclusive, then z/OS treats the program as being authorized. z/OS refers to a program executing with a key between 0 and 7 as executing in system key. While a program is executing in problem program state, z/OS normally restricts it to operating in key 8. Thus, the program cannot alter any storage that is not in key 8. Because it is executing in the problem program state, it cannot use any of the privileged instructions that might be used to change its own state. If required, it can request the operating system to perform actions on its behalf using SVC instructions. As the SVC is such a powerful mechanism, it is important that all programs given control from SVC interrupts are closely controlled. Most of the programs given control from SVC interrupts are part of z/OS itself, and these routines perform operations for the problem program, such as opening a data set or closing a data set.

MVS has a powerful concept know as the Authorized Program Facility (APF). A program can be designated as APF Authorized if it is loaded from an APF authorized library. If this is the case, then the program has the authority to execute a class of SVCs that are themselves APF authorized. If a program that is not APF authorized attempts to execute these SVCs, then the SVC does not perform the requested function. It should be apparent that there must be strict controls over the definitions of programs that are APF authorized. In practice, this control is achieved using a combination of z/OS controls that use SAF and RACF.

Note the following items:

- If a program is in the supervisor state, it can use instructions to change its PSW storage key so that it can make itself APF authorized.
- If a program is running with a PSW that is less than 8, then the z/OS operating system allows it to use MODESET to move to the supervisor state.
- If a program is APF authorized, the z/OS operating system allows it to use MODESET to set any PSW key or change to the supervisor state.

Thus, if any of these three conditions applies, then there is no barrier to obtaining the other privileges.

All normal programs run in Key 8 and problem program mode and are not APF authorized. Therefore, normal problem programs cannot obtain access to any of the privileged instructions that can alter the structure of address spaces, nor can they access data outside of the local address space without prior agreement with the operating system. The operating system is always in control.

If a program has any of the following characteristics, then it should not be possible for the program to gain control in any of those states:

- Not APF-authorized
- Not in the system key
- Not in the supervisor state

The details of the z/OS operating system's usage of storage keys and APF authorization processing can be found in Chapter 7, "z/OS Security", of *Security on the IBM Mainframe*, SG24-7803.

2.1.4 Patch management and the IBM Red Alerts subscription service

It has come to the attention of IBM that its customers might not be validating the currency of their security and system integrity service levels and promptly installing all security and integrity PTFs when they are made available by IBM.

Users of the z/OS operating system should validate the currency of their security and system integrity service levels and take action to promptly install all security and integrity PTFs. Security and system integrity fixes are included in Recommended Service Upgrades (RSUs), and maintaining RSU currency minimizes exposure to security and integrity issues.

Customers should subscribe to the System z Security Portal. The portal enables you to receive the latest critical service information about security and system integrity APARs for z/OS and z/VM. If you are not subscribed to this portal, instructions are located at:

http://www.vm.ibm.com/security/aparinfo.html

Important information about critical security and system integrity APARs and associated fixes are posted to this portal, which can help you plan the timely installation of this critical service. After you are registered at the portal, you receive email notifications of all new posts. More information about the Red Alerts subscription service for z/OS is located at:

http://www.ibm.com/support/docview.wss?uid=swg21290768&wv=1

Validate the currency of your z/OS security and system integrity service levels. Subscribe to the System z Security Portal to receive the latest information about System z security and system integrity service. The timely installation of security and system integrity service is critical in minimizing potential risks and maintaining overall system security and availability.

2.2 Application versus DBMS compliance controls

When working with external auditing and compliance review providers, choosing an approach that includes products and processes that are viewed as an industry recognized best practice can, in many cases, result in high probabilities of obtaining a favorable result from any audit or external review. When information protection approaches are based on application design and homegrown processes, these tend to be viewed in a more critical light, and demand more detailed scrutiny by external audit providers to ensure they provide adequate protection for whatever regulatory or industry initiative in scope.

It is also difficult to construct an application-centric solution that can be implemented and maintained without the direct involvement of privileged users. Application based processes for security further complicate the testing and quality assurance process, as not only will application changes need to be validated for the incremental new business process associated with any change, but regression testing will need to be performed to ensure that the built-in security process has not been tampered with, either by intentional activity or inadvertent modification.

However, there are some practical considerations that can lead to a conclusion that application based controls can be a better decision.For example, the inventory of sensitive information yields a few number of elements needing protection, and the incidence of this data does not justify the significant expenses requiring the acquisition and deployment of sophisticated protection mechanisms.

One example would be where, across a whole organization's DB2 hosted data, there is only a single element contained on one table that would be considered personally identifying information. It would make little sense to deploy OEM provided security solutions, which would be costly to acquire, and would subject the whole DB2 based workload to additional impact. For these situations, an application approach might be a better solution, as it could be isolated to the specific sensitive data element, which would be well identified, and the number of application processes that access this element would be limited in number. But, when taking this approach, it is entirely possible that as organizations mature, through acquisitions, application retirement and modernization, and new business growth, the inventory of sensitive information can grow, and at some juncture, a tipping point will occur that might require a change from application based security to an external security solution approach.

DB2 10 introduces a method of implementing row and column access control as an additional layer of security that you can use to complement the privileges model and to enable DB2 to take part in your efforts to comply with government regulations for security and privacy. You use row and column access control to control SQL application access to your tables at the row level, column level, or both, based on your security policy.

DB2 enforces row and column access control at the table level, transparent to any kind of SQL DML operation or applications that access that table through SQL DML. Existing views do not need to be aware of row or column access controls because the access rules are enforced transparently on its underlying tables and table columns. When a table with row or column level control is accessed through SQL, all users of a table are affected, regardless of how they access the table (through an application, through ad-hoc query tools, through report generation tools, or other access) or what authority or privileges they hold.

2.3 Privileged user controls

Abuse of system access/privileges involves specifically the misuse of bestowed and business justified access to information and systems for unjustified reasons. Once initiated, this abuse can result in an attempt to sell the information for financial gain, a threat of releasing the information in an extortion attempt, or to just inflict damage for various gains. One of the worst aspects of privilege abuse is that while numerous anecdotal examples exist, many others will likely never be discovered.

To ensure the continued health and well-being of any database management system (DBMS), including DB2 and IMS on z/OS, many activities must be performed on a regular basis by system and database administrators. Although these activities can be well controlled by external security processes such as RACF, they are pervasive in effect and can be used in ways that are contrary to security policies.

To cite one possible scenario, suppose there is sensitive data residing on a DB2 table, and the applications that access this table, such as IMS or CICS, are well protected by RACF. The database administrator does not have RACF authority to execute the CICS application, but has database administration authority (DBADM) to administer the table. The database administrator runs an UNLOAD utility against the table, extracting all of the data contained in the table. He or she can then transfer that data through any number of mechanisms to an outside media (FTP, Flash/USB, CSV to spreadsheet, and so on). Because the user has special privileges against the table, there will be no evidence of a security violation that would be reported by RACF.

If, conversely, the environment were protected by an auditing solution, there could be mechanisms that would report on this authorized, but questionable, use of special privileges. A best practice for audit collection is to monitor any SQL or utility access for privileged users. Conversely, you might elect to monitor each utility event or combine looking for one or both classes of events within a time interval. So, although it might be acceptable for the database administrator to access the audited tables during normal business hours, auditing parameters might be set up to look for unusual access patterns outside of normal business hours.

The conundrum is that the nature of these authorities gives the privileged user capabilities to access DB2 and IMS resources and data by means outside the use of the well-protected application environment. This situation has the effect of providing unlimited access to the data from any interface, batch, local, remote, trusted connection, and so on, all the time, thereby circumventing normal transaction-level RACF protection. In a DBMS environment where privileged user authorities have been granted, there must be some mechanism to track and record activities that are performed under the control of these privileged user identifiers.

DB2 10 introduces the concepts of separation of duties and least privilege to address these needs. Separation of duties provides the ability for administrative authorities to be divided across individuals without overlapping responsibilities, so that one user does not possess unlimited authority (for example, SYSADM).

In DB2 10, the SYSTEM DBADM authority allows an administrator to manage all databases in a DB2 subsystem. By default, the SYSTEM DBADM has all the privileges of the DATAACCESS and ACCESSCTRL authorities.

The SECADM authority provides the ability to manage access to tables in DB2 while not holding the privilege to create, alter, drop, or access a table. Furthermore, a DATAACCESS authority can access all user tables without being able to manage security or database objects.

You can use these authorities to minimize the need for the SYSADM authority in day-to-day operations by delegating security administration, system related database administration, and database access duties to the SECADM, SYSTEM DBADM, and DATAACCESS authorities. To go even further, you can separate security administration from the SYSADM authority so that SYSADM can no longer perform security-related tasks.

In addition, in earlier releases of DB2, revocation of administrative privileges granted to specific users triggered a DBMS characteristic referred to as *cascade revocation*. With the use of DB2 native grant/revoke security, the object owner (known as the *grantor*) will in turn grant access privileges to others (known as *grantees*), either to individual IDs, or to RACF controlled group IDs. Based on a change in the privilege grantor business role, some examples might include termination, change in job responsibility, or another event. The desired goal is to remove their access from the security system, and to revoke any administration privileges bestowed on the privilege grantee. However, when the administrative privilege is revoked from the grantor, now known as the *revokee*, privileges granted to others by the revokee are also subject to revocation. This characteristic of cascading revocation of granted privileges is referred to as *cascade revocation* or *cascade revoke*. Because it can be difficult to determine the impact of revocation in large and complex DB2 environments, many installations will simply leave the administration privilege intact, even it the individual is no longer employed. As a result, mature DB2 installations can contain a proliferation of many administrative IDs, with powerful levels of administration authorities, but without any real ongoing business need to exist. As we will discover, DB2 10 provides new system controls that can influence how the revoke mechanism handles cascade revocation.

Privileged users typically need a means of moving data after they have stolen it. Some use corporate or personal email to send it to external parties or accounts. Some smuggle it out on various types of personal devices or media. Others use approved devices, but for unapproved purposes or in an inappropriate manner. While many organizations view control of the avenue of data distribution as a solution, some examples are removal of USB port connections on workstations, restriction on the use of FTP agents, and so on. it is generally easier to control data at the source. The really important message is one of quickly de-provisioning user access and privileges when they are no longer needed. A good rule of thumb is that if you are no longer sending them a paycheck, then do not leave their administration IDs defined in your database systems.

Another issue with privileged user activities on DB2 for z/OS involves the use and control of audit event collection mechanisms. For many customers, DB2 auditing will include the use of the Instrumentation Facility Interface (IFI) audit trace mechanism. This mechanism is controlled by the DB2 Trace facility, which requires special administration privileges. However, in most implementations of an IFI based auditing approach, these trace controls and associated privileges have been granted to the DB2 System Administrators and Database Administration personnel. Indeed, for many customers, these privileged users are tasked with the collection of audit data and the generation of audit reports, the same reports that are then used to verify and report on the activities of these privileged administration users. When DBMS auditing has been implemented in such a manner, it becomes problematic from a separation of roles and responsibilities perspective.

Any mechanism used to audit activities of trusted users must be implemented in such a way as to prevent the privileged user from interfering with the collection of, or contaminating the source of, the audit data. Audit mechanisms against data servers for z/OS must maintain the necessary separation of duties, resulting in assurance of audit data integrity and more accurate reports. These mechanisms allow database administrators to perform their own job duties and allows auditors to run audit reports independently of the database administrators, which results in easier and more accurate audits.

Auditors need to have the ability to adhere to published industry standards and external auditing without relying on the assistance of the personnel being monitored. We will discuss changes with DB2 10 on z/OS that can separate audit event collection controls from privileged database administrators, which can help demonstrate a strengthened roles separation with audit reporting.

2.4 DB2 for z/OS from an evolutionary perspective

In this section, we discuss the evolution of DB2 security. We start with a quick review of DB2 from an historical perspective, with an emphasis on security features. We also take a closer look at the changes brought about by the proliferation of customer facing applications made accessible on the Web and the specific threat of SQL injection.

DB2 Version 1 was delivered in June 1985. As the first version, there were some elementary capabilities that were directly related to security implementation. Security was implemented through the SQL GRANT and REVOKE statement, but without any capability to assign and use secondary authorization IDs. So, in general, the object creator became the owner of the object. Lock duration and lock escalation mechanisms were introduced, which helped maintain data integrity, and provided the ability for different mixtures of workload to coexist, something lacking in previous database management implementations and in CICS/VSAM.

DB2 Version 2, delivered in September 1988, saw some significant enhancements delivered to improve security and audit capabilities. Secondary authorization support was delivered, with improvements in the sign-on and authorization exit processing, which enabled better support for online teleprocessing environments such as CICS and IMS. The DB2 instrumentation facility interface was improved to provide the capability to generate audit trace information for specific events. Also, to provide a mechanism to control the performance characteristics of different types of workload, the resource limit facility (RLF) was introduced.

Some customers began to use RLF as a way to restrict certain types of workload from being processed by DB2. DB2 Version 2 also introduced DBMS enforced referential integrity, which helps better maintain data integrity between tables related by parent/child row values. DB2 also introduced a new allied address space, known as the distributed data facility (DDF) address space. This facility allowed access to data located across multiple DB2 instances. DB2 introduced its strategy for the evolution of distributed relational data base. This was the first real release that supported the basics needed for distributed processing, which as we know, significantly complicates the topic of DB2 security.

DB2 Version 3, introduced in December 1993, saw some incremental improvements in security features. Compressed table space support was introduced: Some customers view compressed data stored on the table space pages as a protection against viewing the data outside of DB2. Archive DB2 recovery log read capability was enhanced through the introduction of the IFI log read exit. This exit enabled customers to process DB2 recovery log records for audit reporting of before and after change images. DB2 continues to improve distributed access capabilities by increasing the number of distributed threads supported by DB2 to 10,000, and the introduction of query I/O parallelism makes the introduction of query intensive, decision support applications feasible.

DB2 Version 4, introduced in November 1995, significantly improved support for distributed workloads. Customers were designing and deploying workstation based applications, and DB2 provided additional improvements to support even larger number of distributed connections. DB2 Version 4 introduced data sharing to allow customers who were constrained by the limitations of virtual storage or number of concurrent processing requests to grow horizontally.

Query parallelism performance was improved, resulting in even more dynamic SQL from distributed requestors, and more demands for authority to access data for ad-hoc query requests. The RACF/DB2 interface is delivered, allowing customers to define and control access to DB2 resources from within the external security manager.

DB2 Version 5, introduced in June 1997, saw a number of large customers implement web-enabled customer facing applications for the first time. Because of this action, traditional windows for routine database maintenance were shrunk or eliminated entirely. Online utilities were first introduced. DB2 provided direct connectivity through TCP/IP along with increased connectivity options for workstation clients. DB2 with IBM DB2 Connect™, introduced the ability to perform 56-bit single DES encryption on ID and password flows. The RACF/DB2 interface support, delivered in DB2 Version 4, was improved to provide protection for additional resource types.

DB2 Version 6, introduced in June 1999, provided extensions to the DB2 identification fields for use by distributed applications, known as the extended workstation identification fields. The SQLESETI API was introduced to work with these extended identifiers.

DB2 Version 8, introduced in March 2004, provided the ability to perform TDES encryption on selected columns with the use of SQL extension ENCRYPT and DECRYPT. Additional synergy is seen with RACF with the introduction of multilevel security or MLS, through the SECLABEL SQL parameter. DB2 Version 8 received EAL3 certification under the Common Criteria.

DB2 Version 9, introduced in June 2007, provided support for the Secure Sockets Layer (SSL) protocol by implementing the z/OS Communications Server IP Application Transparent Transport Layer Security (AT-TLS) function. SSL encryption has been available on z/OS for a long time, but with AT-TLS, more applications will be able to offer this level of encryption. DB2 9 for z/OS makes use of this new facility and now offers SSL encryption using a new secure port.

DB2 9 introduced the use of roles and network trusted contexts. Support for trusted context provides the ability within a specific trusted context for a DB2 authorization ID to acquire a special set of privileges that are not available outside of that trusted context by defining roles. A role is a database entity that groups together one or more privileges and can be assigned to users. A role provides privileges, in addition to the current set of privileges that are granted to the primary and secondary authorization identifiers. A role can own objects if the objects are created in a trusted context with the role defined as the owner.

DB2 9 works with the IBM enterprise identity mapping solution. Enterprise identity mapping (EIM) enables the mapping of user identities across servers that are integrated but that do not share user registries. DB2 supports the EIM capability by implementing the SAF user mapping plug-in callable service, which is part of the z/OS Security Server (RACF). EIM actually defines associations between an EIM identifier and user IDs in registries that are part of OS platforms, applications, and middleware. Applications, typically servers, can then use an EIM API to find a mapping that transforms the installation-level user ID initially used for authentication to a local user ID, which can in turn be used to access local resources.

DB2 9 allows the application server to propagate user IDs to DB2, uses Enterprise Identity Mapping for improved DB2 and RACF auditing, uses ROLEs for access control and has the objects owned by ROLES, and finally uses special registers to manage object resolution instead of SQLIDs. New filtering options on the START TRACE command give you the ability to select your trace records more carefully, thus minimizing the amount of data collected and monitoring more closely access from specific users or locations.

The new possibilities include:

- Roles (used through a trusted context) can be filtered through the new keyword ROLE.
- Other trace filter capabilities for package name, collection name, and so on.
- New keywords are introduced to provide exclude trace filters.
- The ability to use wild cards, rather than the full names.

Security, regulatory compliance, and audit capability improvements are included in DB2 10. Enhanced security extends the role-based model introduced in DB2 9.

DB2 10 provides granular authorities and system parameters that separate data access from the administration of security, application, database, and system.

DB2 10 provides administration flexibility for specific security role settings, preventing data access exposure to unauthorized applications or administrators. This role-based security model, combined with the application based row access control and column masking of sensitive information, enhances the secure database environment for your business.

In addition, you can define and create different audit policies to address the various security needs of your business.

All of these features provide tighter controls, allow more security flexibility, and provide tremendous regulatory and audit compliance capabilities.

Part 2

DB2 capabilities

In this part, we describe the main functions that contribute to security in a DB2 for z/OS environment.

First, we investigate the synergy of DB2 for z/OS with the installed external security manager, RACF. We then describe the DB2 for z/OS security functions, concentrating on the enhancements of DB2 10 for z/OS. We also discuss the usage of the cryptographic ecosystem available on z/OS and System z processors and the new DB2 10 functions related to user authentication and audit policies.

This part contains the following chapters:

3

Administrative authorities and security-related objects

Within DB2, privileges are grouped into administrative authorities and each authority is vested with a specific set of privileges.

Security is optimal when each function can be broken down to a level of least privilege and users have access to the privileges required to perform all of their assigned functions.

DB2 10 implements new administrative authorities and privileges, as well as new security system parameters that are designed to help businesses comply with government regulations and to simplify the management of authorities.

The new authorities provide a separation of duties to different users without overlapping the responsibilities of each authority. SECADM and SYSTEM DBADM[1] are the new authorities that minimize the need of SYSADM authority in day-to-day operations and also to help satisfy the auditors. Implementing some of these features may require a transfer of job responsibilities and new job roles.

In addition, to make sure that administrative authorities are not misused, DB2 10 provides a new capability to audit the administrative authorities. You can define and create different audit policies to address the different security needs of your business. New auditing features include the ability to dynamically enable the auditing of all static and dynamic statements against a table, and the ability to audit any use of a system or database authority. These new audit functions are described in Chapter 8, “Audit policies” on page 151.

In this chapter, we introduce the new administrative authorities, the separate security features, and the concept of security-related objects. We provide the rationale and show basic examples for use of the new authorities.

We describe the framework for building the right security setup at your installation. There is not one solution or one correct way to implement DB2 security on the System z platform. There are a number of different elements and they can be combined in various ways.

[1] The already existing database DBADM has not changed.

This chapter contains the following topics:

- Rationale for new features
- Management of security-related objects
- SECADM
- SYSTEM DBADM
- ACCESSCTRL
- DATAACCESS
- Reassigning powerful privileges held by SYSADM and SYSCTRL
- Revoking without cascade
- Debugging and performance analysis privileges
- DSNZPARMs related to security

The chapters in Part 3, “Implementation scenarios” on page 209 provide scenarios about how a company could combine and make use of the new features.

3.1 Rationale for new features

More data is available to more people through more channels. The cost of a data breach is significant and customers are asking for a wide range of enhancements for security.

DBAs automatically have access to data in all tables they create. System administrators have the ability to see all data in all tables in the subsystem, as well as the ability to GRANT access to any object in the subsystem. Such accounts are granted privileges that allow actions that can have a great impact on database security and operation. It is especially important to grant access to privileged accounts only to those persons who are qualified and authorized to use them. Information needs to be classified and made available only to those who really need it.

In today’s world, there is a growing demand to limit privileges to the minimum required to perform a job function. The principle of *least privilege* needs to be enforced, that is, only permit users to do what they really need to do, and minimize overlap. A separation of access control from administrative duties is warranted and justifiable. Such a separation reduces the potential security risk associated with general availability of powerful privileges. Users granted privileges to perform unauthorized actions may do so either intentionally or unintentionally.

The SYSCTRL authority is intended to separate system control functions from administrative functions. However, SYSCTRL is not a complete solution for a high-security system. If any plans have their EXECUTE privilege granted to the PUBLIC role, an ID or role with the SYSCTRL authority can grant itself the SYSADM authority. The only control over such actions is to audit the activity of IDs with high levels of authority. SYSADM authority allows an ID to have control over multiple phases of a task.

In DB2 10, administrative tasks can now be split into multiple auditable authorities.The granularity and flexibility in the new DB2 10 administrative authorities help you achieve adequate separation of duties and responsibilities and prevent a single user from possessing unlimited privileges.

Security tasks can now be separated from system tasks. Sensitive data can be protected from privileged users. SQL tuning can be separated from data access. Access control can be transferred from system administrators to security administrators. The new audit policy functionality allows you to audit users with powerful authorities and single tables or groups of tables.

In this section, we discuss:

- Administrative authorities available in DB2 9
- New administrative authorities available in DB2 10

3.1.1 Administrative authorities available in DB2 9

Before we discuss the new authorities, let us take a quick look at the existing authorities. You can control access within DB2 by granting or revoking privileges and related authorities that you assign to authorization IDs or roles. A privilege enables its holder to perform a specific operation. Privileges can be explicit or implicit. An explicit privilege is a specific type of privilege. An implicit privilege comes from the ownership of objects, including plans and packages. Existing system authorities in DB2 9 are:

- Install SYSADM

  Install SYSADM authority is assigned to one or two IDs when DB2 is installed; it cannot be assigned to a role. These IDs have all the privileges of the SYSADM authority.

- SYSADM

  The SYSADM authority includes all the privileges, including system privileges, for creating objects and accessing all data. Depending on the setting of the SEPARATE_SECURITY system parameter, the SYSADM authority can also create security objects and grant and revoke privileges.

- DBADM

  The DBADM authority includes the DBCTRL privileges over a specific database. A user with the DBADM authority can access any tables in a specific database by using SQL statements.

- DBCTRL

  The DBCTRL authority includes the DBMAINT privileges on a specific database. A user with the DBCTRL authority can run utilities that can change the data.

- DBMAINT

  A user with the DBMAINT authority can grant the privileges on a specific database to an ID or role.

- SYSCTRL

  The SYSCTRL authority is designed for administering a system that contains sensitive data. With the SYSCTRL authority, you have nearly complete control of the DB2 subsystem. However, you cannot access user data directly unless you are explicitly granted the privileges to do so.

- PACKADM

  The PACKADM authority has the package privileges on all packages in specific collections and the CREATE IN privilege on these collections.

- Installation SYSOPR

  Installation SYSOPR authority is assigned to one or two IDs when DB2 is installed; it cannot be assigned to a role. These IDs have all the privileges of the SYSOPR authority.

- SYSOPR

  A user with the SYSOPR authority can issue all DB2 commands except ARCHIVE LOG, START DATABASE, STOP DATABASE, and RECOVER BSDS.

3.1.2 New administrative authorities available in DB2 10

The new granular system authorities in DB2 10 are:

- SECADM

  The SECADM authority enables you to manage security-related objects in DB2 and control access to all database resources. It does not have any inherent privilege to access data stored in the objects, such as tables.

- SYSTEM DBADM

  The SYSTEM DBADM authority allows an administrator, an authorization ID or a role, to manage databases across a DB2 subsystem, while optionally having no access to the data in the databases. In other words, the SYSTEM DBADM authority enables you to create, alter, and drop DB2 objects and issue commands for a DB2 subsystem, but does not give you the authority to access the data or the ability to grant or revoke privileges. When an ID with SYSTEM DBADM without DATAACCESS creates a table, it has access to the table contents because ownership overrides ‘without DATAACCESS’.

- ACCESSCTRL

  The ACCESSCTRL authority allows you to grant explicit privileges to authorization IDs or roles by issuing SQL GRANT statements. It enables you to grant privileges on all objects and resources, except the CREATE_SECURE_OBJECT privilege and the SYSTEM DBADM, DATAACCESS, and ACCESSCTRL authorities.

- DATAACCESS

  The DATAACCESS authority allows you to access and update data in user tables, views, and materialized query tables in a DB2 subsystem. It also allows you to execute plans, packages, functions, and procedures.

- SQLADM

  The SQLADM authority allows you to issue the SQL EXPLAIN statements, execute the PROFILE commands, run the RUNSTATS and MODIFY STATISTICS utilities on all user databases, and execute system-defined routines, such as stored procedures or functions, and any packages that are executed within the routines.

3.1.3 New system privileges in DB2 10

There are two new privileges in DB2 10: EXPLAIN and CREATE_SECURE_OBJECT. For details about all the DB2 system privileges, see Chapter 5, “Managing access through authorization IDs and roles”, of *DB2 10 for z/OS Administration Guide*, SC19-2968.

- EXPLAIN

  The EXPLAIN system privilege provides you the ability to issue EXPLAIN, PREPARE, and DESCRIBE statements without requiring the privilege to execute the statements. It allows you to validate your applications and SQL syntaxes, without requiring access to data, before putting them into production.

  - The SQL EXPLAIN PLAN and EXPLAIN ALL statements issue the statements without requiring additional privileges.
  - The SQL PREPARE and DESCRIBE TABLE statements prepare and describe the statements without requiring additional privileges on the object.
  - The BIND command allow users to specify the EXPLAIN(ONLY) and SQLERROR(CHECK) options without creating a plan or package.

  – Dynamic SQL statements that have the special register CURRENT EXPLAIN MODE set to EXPLAIN allow the capture of information about the statements without executing them.

  An authorization ID or role with any of the following authorities or privileges can grant the EXPLAIN privilege:

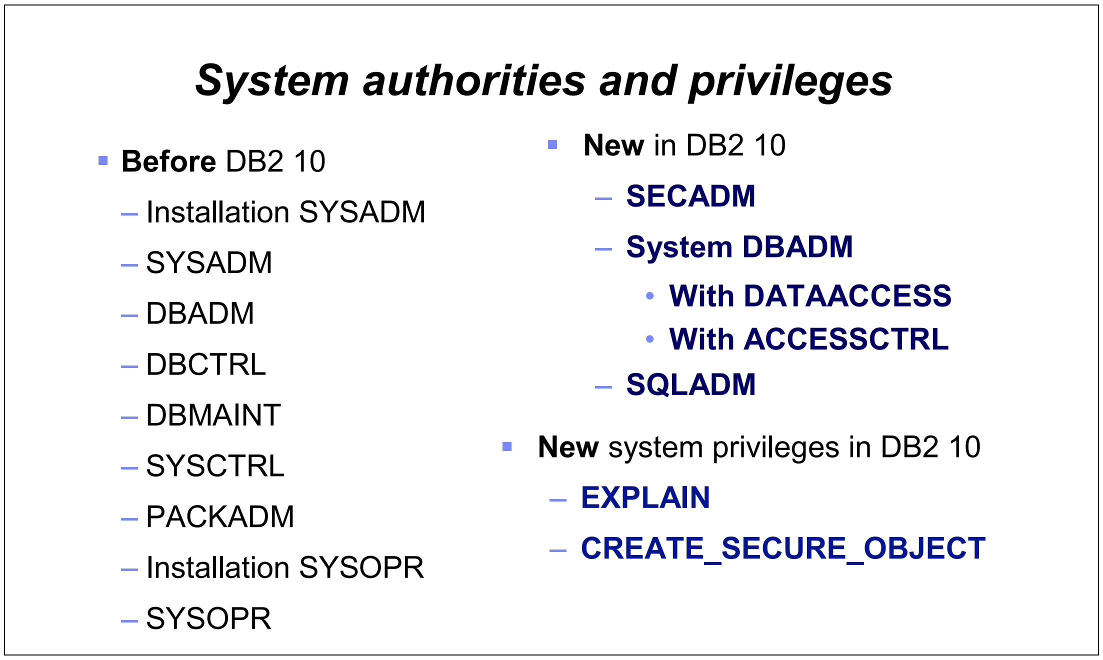
  – The SECADM authority.
  – The ACCESSCTRL authority.
  – The SYSADM authority if the SEPARATE SECURITY system parameter is set to NO at the installation.
  – The EXPLAIN privilege with the WITH GRANT OPTION.

- CREATE_SECURE_OBJECT

  The CREATE and ALTER statements create secure objects, such as a secure trigger or a user-defined function. If a trigger is defined for tables that are enforced with row or column access control, it must be secure. If a user-defined function is referenced in the definition of a row permission or column mask, it must be secure. In addition, if a user-defined function is invoked in a query and its arguments reference columns with column masks, the user-defined function must be secure.

3.1.4 Current administrative authorities and their privileges

In Figure 3-1, we show the old and new authorities and the new privileges. When you migrate to the new DB2 authorities, the existing SYSADM and DBADM privileges are the most affected.



System authorities and privileges

- **Before** DB2 10
  – Installation SYSADM
  – SYSADM
  – DBADM
  – DBCTRL
  – DBMAINT
  – SYSCTRL
  – PACKADM
  – Installation SYSOPR
  – SYSOPR
- **New** in DB2 10
  – **SECADM**
  – **System DBADM**
    • **With DATAACCESS**
    • **With ACCESSCTRL**
  – **SQLADM**
- **New** system privileges in DB2 10
  – **EXPLAIN**
  – **CREATE_SECURE_OBJECT**

Figure 3-1 New DB2 10 authorities and privileges

Changed system authority in DB2 10: With separate security NO, SYSADM includes these authorities:

- SECADM, SYSTEM DBADM, SQLADM.
- ACCESSCTRL, DATAACCESS.
- SYSADM also includes the new EXPLAIN and CREATE_SECURE_OBJECT privileges.

Privileges are grouped into administrative authorities, and each administrative authority has a specific set of privileges. Here is the list all of the DB2 for z/OS administrative authorities and the grantable privileges that each of them has:

- ACCESSCTRL
  - Included authorities: None
  - Additional grantable privileges:
    - Privileges on all catalog tables: SELECT
    - Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES): DELETE, INSERT, and UPDATE
    - Privileges on security: GRANT and REVOKE
- DATAACCESS
  - Included authorities: None
  - Additional grantable privileges:
    - System privileges: DEBUGSESSION
    - Privileges on all user tables, views, and MQTs: DELETE, INSERT, SELECT, and UPDATE
    - Privileges on all plans, packages, and routines: EXECUTE
    - Privileges on all user databases: LOAD, RECOVERDB, REORG, and REPAIR
    - Privileges on all JARs: USAGE
    - Privileges on all sequences: USAGE
    - Privileges on all distinct types: USAGE
    - Privileges on all catalog tables: SELECT
    - Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES): DELETE, INSERT, and UPDATE
- DBADM
  - Included authorities: DBCTRL, DBMAINT
  - Additional grantable privileges are privileges on tables in a database: ALTER, DELETE, INDEX, INSERT, REFERENCES, SELECT, TRIGGER, and UPDATE
- DBCTRL
  - Included authorities: DBMAINT
  - Additional grantable privileges are privileges on a database: DROP, LOAD, RECOVERDB, REORG, and REPAIR
- DBMAINT
  - Included authorities: None
  - Additional grantable privileges are privileges on a database: CREATETAB, CREATETS, DISPLAYDB, IMAGCOPY, STATS, STARTDB, and STOPDB

- Install SYSADM
  - Included authorities: SYSADM, SYSCTRL, DBADM, installation SYSOPR, SYSOPR, PACKADM, DBCTRL, DBMAINT
  - Additional grantable privileges are privileges on security: GRANT and REVOKE
- Install SYSOPR
  - Included authorities: SYSOPR
  - Additional grantable privileges: ARCHIVE and STARTDB (cannot alter access mode)
- PACKADM
  - Included authorities: None
  - Additional grantable privileges:
    - Privileges on a collection: CREATEIN
    - Privileges on all packages in a collection: BIND, COPY, and EXECUTE
- SECADM
  - Included authorities: ACCESSCTRL
  - Additional grantable privileges:
    - Privileges on all catalog tables: SELECT
    - Privileges on all updatable catalog tables: DELETE, INSERT, and UPDATE
    - Privileges on security: GRANT and REVOKE
    - Privileges on security-related objects: ALTER, CREATE, and DROP
- SQLADM
  - Included authorities: None
  - Additional grantable privileges:
    - System privileges: EXPLAIN, MONITOR1, and MONITOR2
    - Privileges on all catalog tables: SELECT
    - Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES): DELETE, INSERT, and UPDATE
- SYSADM
  - Included authorities: SYSCTRL, DBADM, installation SYSOPR, SYSOPR, PACKADM, DBCTRL, and DBMAINT
  - Additional grantable privileges:
    - Privileges on all plans: EXECUTE
    - Privileges on all routines: EXECUTE
    - Privileges on all packages: All privileges
    - Privileges on distinct types: USAGE
    - Privileges on sequences: USAGE
    - System privileges: DEBUGSESSION

- SYSCTRL
  - Included authorities: Installation SYSOPR, SYSOPR, DBCTRL, DBMAINT, ACCESSCTRL (except the ability to grant certain authorities, such as DBADM, SYSADM, PACKADM, and certain privileges, such as DELETE, INSERT, SELECT, and UPDATE on user tables or views, EXECUTE on plans, packages, functions, or stored procedures, PACKADM on collections, and USAGE on distinct types, JARs, and sequences)
  - Additional grantable privileges:
    - System privileges: BINDADD, BINDAGENT, DBDS, CREATEALIAS, CREATEDBA, CREATEDBC, CREATESG, CREATETMTAB, MONITOR1, MONITOR2, and STOSPACE
    - Privileges on all tables: ALTER INDEX and REFERENCES TRIGGER
    - Privileges on all catalog tables: DELETE, INSERT, SELECT, and UPDATE
    - Privileges on all plans: BIND
    - Privileges on all packages: BIND and COPY
    - Privileges on all packages: CREATEIN
    - Privileges on all schemas: ALTERIN, CREATEIN, and DROPIN
    - Privileges on use: BUFFERPOOLS, STOGROUP, and TABLESPACE
- SYSOPR
  - Included authorities: None
  - Additional grantable privileges:
    - Privileges: DISPLAY, RECOVER, STOPALL, and TRACE
    - Privileges on routines: DISPLAY, START, and STOP
- SYSTEM DBADM
  - Included authorities: SQLADM.
  - Additional grantable privileges:
    - System privileges:

      BINDADD, BINDAGENT, CREATEALIAS, CREATEDBA, CREATEDBC. CREATETMTAB, DISPLAY, EXPLAIN, MONITOR1, MONITOR2, SQLADM, STOPALL, and TRACE
    - Privileges on all collections: CREATEIN
    - Privileges on all user databases: CREATETAB, CREATETS, DISPLAYDB, DROP, IMAGCOPY, RECOVERDB, STARTDB, and STOPDB
    - Privileges on all user tables (except for those defined with row permissions or column masks): ALTER, INDEX, REFERENCES, and TRIGGER
    - Privileges on all packages: BIND COPY
    - Privileges on all plans: BIND
    - Privileges on all schemas: ALTERIN, CREATEIN, and DROPIN
    - Privileges on all sequences: ALTER
    - Privileges on all distinct types: USAGE
    - Privileges on use: TABLESPACE

- Privileges on all catalog tables: SELECT
- Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES): DELETE, INSERT, and UPDATE

3.2 Management of security-related objects

DB2 10 introduces the concept of security-related objects. When separate security is set to No, it is business as usual and any ID with SYSADM authority can manage the security related objects.

When separate security is set to Yes, only an ID with SECADM authority can manage the security related objects.

The security related objects are:

- Role
- Trusted context
- Row permission
- Column mask

Typically, there are more people with the powerful SYSADM authority than is required. Separating security administration from system administration is a step forward towards improved security.

For example, with separate security set to Yes, a user with the SYSADM authority is no longer able to assume another identity using SET CURRENT SQLID, create a trusted context to take over another person's authority, or to see through a column mask or row permission.

3.3 SECADM

DB2 System Administrators and many DB2 DBAs typically have the SYSADM authority, which provides full privileges to all objects within the subsystem. Too many users have SYSADM. There is a growing demand to minimize the use of SYSADM. Sometimes an authid with the SYSADM authority is shared amongst multiple people. DB2 System Administrators typically do not need to have full access to all the tables in the subsystem.

Good security practice demands both the separation of duties and the assignment of least privilege. A different group should manage access to data other than the owner of the data. Management of security objects like Trusted Contexts and Roles should be restricted to a few people and not be made available to administrators with SYSADM or SYCTRL. For example, with separate security, SYSADMs can still create a trusted context and assign themselves a ROLE or allow themselves to perform actions using someone else's AUTHID.

The new SECADM authority addresses these issues.The SECADM authority can manage security-related objects in DB2 and control access to all database resources. A user with SECADM authority cannot create, alter, or drop tables. The authority cannot access any user data. Implementing a separate SECADM authority prevents SYSADM and SYSCTRL authorities from granting or revoking privileges.

A new DSNZPARM SEPARATE_SECURITY authority is now available to separate security administration, database administration, and data access control from system administration.

Separating the SYSADM authority (a combination of security and system administration) can help you simplify your system administration and strengthen the security administration of your business data. For example the new SECADM authority provides a user with the ability to manage access to a table in DB2, but that user cannot create, alter, or drop the table. Also, that user cannot access the data within the table.

Removing the SECADM authority through a DSNZPARM change does not revoke any dependent privileges that were granted by SECADM.

3.3.1 Privileges held by the SECADM authority

In Table 3-1 shows the privileges held by the SECADM authority.

Table 3-1 Privileges held by SECADM

Privilege category	Description
SELECT	Select all catalog tables.
INSERT, UPDATE, DELETE	All updatable catalog tables, including SYSIBM.SYSAUDITPOLICIES.
GRANT, REVOKE (SQL DCL)	▶ Grant access to and revoke access from all database resources. ▶ Revoke explicit privileges that are granted by SECADM. ▶ Revoke explicit privileges granted by others using the BY clause.
CREATE, ALTER, DROP, COMMENT	▶ Column mask controls. ▶ Row permission controls.
Alter user table to	▶ Activate and deactivate column level access control. ▶ Activate and deactivate row level acces control.
CREATE, DROP, COMMENT	Role.
CREATE, DROP, COMMENT, ALTER	Trusted context.
Online DSNZPARM changes	SECADM can perform any online DSNZPARM change.
Utilities	▶ Unload catalog tables only. ▶ DSN1SDMP.
Commands	Start, alter, stop, and display trace.
CREATE_SECURE_OBJECT	Alter the SECURED or NOT SECURED clause on triggers and user-defined functions.

3.3.2 A new separate security install parameter

A new DSNZPARM SEPARATE_SECURITY authority is now available to separate security administration from system administration. At a high level, if you use the DEFAULT value of NO, the SYSADM authority behaves the same as it did in DB2 9. Regardless of the SEPARATE_SECURITY setting, SYSADM has SELECT privileges on all catalog tables and DELETE/INSERT/UPDATE privileges on all updatable catalog tables, except SYSAUDITPOLICIES.

The privileges implicitly held by the new SECADM authority are:

- DELETE/INSERT/UPDATE on SYSIBM.SYSAUDITPOLICIES
- GRANT and REVOKE
- ALTER CREATE DROP on security-related objects
- CREATE_SECURE_OBJECT
- SELECT on all catalog tables

Regardless of the SEPARATE_SECURITY setting, SYSCTRL has all the catalog table privileges and the implicitly held privilege for GRANT and REVOKE.

SEPARATE_SECURITY set to YES

If you set SEPARATE_SECURITY to YES, the SYSADM authority can no longer manage security-related objects (that is, roles, trusted contexts, row permissions, and column masks) or have the ability to grant or revoke privileges that are granted by others. The SYSCTRL authority also cannot manage roles or grant or revoke privileges that are granted by others. Instead, the SECADM authority manages all security-related objects. The SECADM and ACCESSCTRL authorities control access to all databases even though they cannot access any user data in the databases.

In addition, the SYSADM authority can only set CURRENT SQLID on its primary or one of its secondary authorization IDs. The SYSADM, SYSCTRL, and system DBADM authorities can only set BIND OWNER on the primary or one of the secondary authorization IDs of the binder. Finally, the SYSADM authority has no implicit insert, update, or delete access to the SYSIBM.SYSAUDITPOLICIES table.

SEPARATE_SECURITY is set to NO (default)

If you set SEPARATE_SECURITY to NO, the SYSADM authority retains all the existing privileges and responsibilities and gets implicit privileges of the SECADM authority. In other words, the SYSADM authority continues to be the security administrator and manage all security-related objects, perform grants, and revoke privileges that are granted by others. In addition, it obtains implicit insert, update, delete access on the SYSIBM.SYSAUDITPOLICIES table and is able to set CURRENT SQLID and BIND OWNER to any value.

Setting SEPARATE_SECURITY to NO also allows the SYSCTRL authority to obtain most of the implicit privileges of the ACCESSCTRL authority. SYSCTRL can manage roles, perform certain grants, revoke privileges that are granted by others, and set BIND OWNER to any value.

3.3.3 Migration to DB2 10 and SEPARATE_SECURITY YES / NO

Note: The installation SYSADM authority is not affected by the setting of SEPARATE_SECURITY DSNZPARM. It can always perform security related tasks.

Tip: Before setting the SEPARATE_SECURITY field to YES, set at least one SECADM system parameter to an authorization ID, or create the necessary trusted contexts and roles. If you specify YES for SEPARATE_SECURITY, the system administrator authority cannot be used to perform security tasks, and the SECADM authority is required to manage trusted contexts and roles. If both SECADM system parameters are set to roles and those roles have not been created, only SYSADM has the authority to manage trusted contexts and roles.

Removing the SECADM authority through a DSNZPARM change does not revoke any dependent privileges that were granted by SECADM.

Be aware that the default for DSNZPARM SECADM1 is SECADM and the default for SEACDM1_TYPE is an authid. These values are also the defaults for SECADM2.

Note: Regarding SECADM setup, if SEPARATE_SECURITY is set to NO, the SYSADM authority implicitly holds the SECADM authority. If you do not provide your choices for SECADM1 and SECADM2 to DB2, the default is an authid named SECADM for both. If a RACF group or authid is created with this name, it is possible for someone to acquire the SECADM authority

If you are intending to use SEPARATE_SECURITY YES, consider carefully which RACF groups or roles hold the SECADM authority. Consider setting the SECADM1 and SECADM2 to those values and start DB2 with SEPARATE_SECURITY NO. When DB2 starts, those authids holding SECADM can manage security-related objects, and authids holding the SYSADM authority can manage security-related objects.

After you are confident that the authids holding SECADM can take full control of the security-related objects, you should switch to SEPARATE_SECURITY YES. Such an approach provides a transition period as opposed to switching to YES when you use DB2 10. Remember that two of the security-related objects are new with DB2 10: column masks and row permissions.

3.3.4 Role naming considerations

Role names cannot be PUBLIC, ACCESSCTRL, DATAACCESS, DBADM, DBCTRL, DBMAINT, NONE, NULL, PACKADM, SECADM, SQLADM, or any name beginning with “SYS.”

3.4 SYSTEM DBADM

In many installations, DBAs are granted full privileges on all the databases in a subsystem, including the ability to configure the subsystem, create database objects, and manage permissions on those objects. When A DBA creates database objects, the creator AUTHID acquires ownership of the objects. With ownership comes the ability for the DBA to assign access to users, ROLEs, or RACF groups.

The privilege of the object owner to define access to the owned objects is referred to as Discretionary Access Control (DAC) and is the most common access control method found in popular DBMSs. Thus, any assignment of object creation privileges within the database needs to be carefully considered for security implications.

For example, when an object owner grants SELECT access to a table using the WITH GRANT option to and AUTHID, that AUTHID now can grant the SELECT privilege to other users.

From a separation of duties and security viewpoint, the privilege to assign permissions to others should be assigned only to authorized administrators, which promotes management control of authorizations and aids in the administration of privilege assignments.

DB2 10 introduces the SYSTEM DBADM authority, which has the following capabilities:

- Manages resources in all databases.
- Does not have access to data or the ability to grant and revoke privileges.
- Executes system-defined routines (that is, stored procedures or functions) and any packages within the routines.
- Has implicit SELECT access on all catalog tables.

The SYSTEM DBADM authority allows an administrator to manage all databases in a DB2 subsystem. By default, the SYSTEM DBADM has all the privileges of the DATAACCESS and ACCESSCTRL authorities. If you do not want a user (an authorization ID or role) with the SYSTEM DBADM authority to grant any explicit privileges, you can specify the WITHOUT ACCESSCTRL clause in the GRANT statement when you grant the authority. If you do not want a user with the SYSTEM DBADM authority to access any user data in the databases, you can specify the WITHOUT DATAACCESS clause in the GRANT statement when you grant the authority.

Using the following statement, we establish an ID that can manage all objects with no access to data and no ability to grant or revoke:

```
GRANT DBADM WITHOUT ACCESSCTRL WITHOUT DATAACCESS ON SYSTEM TO PROD_DBAS;
```

PAOLOR7 connects to RACF group PROD_DBAS. User ID PAOLOR7 can now manage all the databases in the subsystem. If necessary, you can still grant explicit privileges (that is, SELECT) to RACF group PROD_DBAS to access data or perform grants.

From an auditing point of view, it is important to realize that an authid holding the SYSTEM DBADM authority WITHOUT DATAACCESS is still:

- Able to select the data in all tables created with the SYSTEM DBADM authority because the authid owns the objects
- Able to receive privileges directly, for example, grants from other users
- Able to exercise all privileges available through the RACF groups to which this authid belongs.

3.5 ACCESSCTRL

Management of access control on DB2 objects can now be separated from system administration and database administration. ACCESSCTRL allows for object access management without requiring the ownership of an object or powerful authorities like SYSADM.

The ACCESSCTRL authority has the following capabilities:

- Grants privileges on all but security-related objects and resources.
- Revokes privileges on all but security-related objects and resources that are granted by itself or others.
- Does not grant the SYSTEM DBADM, DATAACCESS, or ACCESSCTRL authority.
- Has implicit SELECT access on all catalog tables.

When SEPARATE_SECURITY is NO, the SYSADM, SYSCTRL, and SECADM authorities have ACCESSCTRL implicitly. When SEPARATE_SECURITY is YES, the SECADM authority has ACCESSCTRL implicitly.

Using a user ID holding the SYSADM authority and with SEPARATE_SECURITY set to NO, we execute the following SQL:

```
GRANT ACCESSCTRL ON SYSTEM TO PROD_ACCESS;
```

The people connected to this RACF group control all access to production databases.

PAOLOR8 connects to RACF group PROD_ACCESS. PAOLOR8 can now manage access to all objects. When the person using AUTHID PAOLOR8 leaves the company, the revocation of ACCESSCTRL does not allow the revocation of dependent privileges.

The following statement shows the mandatory clause of the REVOKE statement:

```
REVOKE  ACCESSCTRL ON SYSTEM FROM PROD_ACCCESS NOT INCLUDING DEPENDENT PRIVILEGES;
```

3.6 DATAACCESS

System administrators and DBAs typically hold authorities that allow them to see all data in user tables. In an environment with SEPARATE_SECURITY=YES, the authority to see all user data can be better controlled. DATAACCESS can be granted to a small number of people that really need it, and such access can be made unavailable to DBAs.

The DATAACCESS authority has the following capabilities:

- Has the ability to access data in all user tables, views, and materialized query tables.
- Has the ability to execute all plans, packages, functions, and procedures.
- Has implicit SELECT access on all catalog tables.

Using a user ID holding the SYSADM authority and with SEPARATE_SECURITY set to NO, we execute the following SQL:

```
GRANT DATAACCESS ON SYSTEM TO PROD_DATA;
```

PAOLOR9 connects to RACF group PROD_DATA. PAOLOR9 now can see content of all tables in the subsystem. This authority may be required by people doing data profiling. This authority should be made available to only a few people, because it enables access to all data in user tables, views, and materialized query tables in the DB2 subsystem.

For example, this authority could also be made available temporarily to:

- Performance people needing to see data content to improve performance
- Auditors verifying data content changes
- Production DBAs needing to run implementation steps to unload and load data
- A developer investigating a production problem

The following statement revokes DATAACCESS system authority. Note that dependent privileges cannot be revoked.

```
REVOKE DATAACCESS ON SYSTEM FROM PROD_DATA NOT INCLUDING DEPENDENT PRIVILEGES;
```

3.7 Reassigning powerful privileges held by SYSADM and SYSCTRL

We describe some high level scenarios to increase separation of duties using some of the new authorities.

When you choose a configuration in your environment, consider keeping the same granularity in development and production, which allows for smoother deliveries into production.

3.7.1 Maintaining a existing common model

This scenario has no separation of duties. We show a common authority setup in Example 3-1.

Example 3-1 Common authority setup

```
SEPARATE_SECURITY=NO (DSNZPARM)
GRANT SYSADM TO PROD_SYSADM;
```

JOHN and SALLY are connected to RACF group PROD_SYSADM. They implicitly have the SECADM authority and can manage security-related objects. The database administrators are also connected to the RACF PROD_SYSADM.

3.7.2 Separating security administration from system administration

This scenario has more separation. We set SEPARATE_SECURITY DSNZPARM to YES and we establish separate security by assigning SECADM1=SAMANTHA with SECADM1_TYPE=AUTHID and SECADM2=OFFICER with SECADM2_TYPE=ROLE.

SAMANTHA is an AUTHID with access to TSO and DB2. ENFORCER is a ROLE that is available to three people through a trusted context. Example 3-2 shows the DDL for this scenario.

Example 3-2 Sample DDL

```
CREATE ROLE OFFICER;
CREATE TRUSTED CONTEXT SEC_OFFICER_MARILYN
BASED UPON CONNECTION USING SYSTEM AUTHID MARILYN
ATTRIBUTES (JOBNAME 'MARILYN')
DEFAULT ROLE OFFICER
ENABLE;
CREATE TRUSTED CONTEXT SEC_OFFICER_KATE
BASED UPON CONNECTION USING SYSTEM AUTHID KATE
ATTRIBUTES (JOBNAME 'KATE')
DEFAULT ROLE OFFICER
ENABLE;
CREATE TRUSTED CONTEXT SEC_OFFICER_WILLIAM
BASED UPON CONNECTION USING SYSTEM AUTHID WILLIAM
```

```
ATTRIBUTES (JOBNAME 'WILLIAM')
DEFAULT ROLE OFFICER
ENABLE;
```

In Example 3-3, we show the setup for the system administrators.

Example 3-3 Administrator setup

```
GRANT SYSADM TO PROD_SYSADM;
```

JOHN and SALLY are connected to RACF group PROD_SYSADM. They can no longer manage access control, audit policies, or security-related objects, including roles and trusted contexts. John and Sally cannot grant or revoke privileges that are granted by others.

Kate, Marilyn and William acquire the OFFICER role when they log onto TSO and obtain the SECADM authority.

3.7.3 Separating the system database administration from system and security administration

In this scenario, we show more separation of duties. The SYSTEM DBADM authority can be set to manage all objects across the DB2 subsystem, with the ability to access the data and grant/revoke privileges on the DB2 objects. The SYSTEM DBADM authority allows you to manage objects in all databases

We set SEPARATE_SECURITY DSNZPARM to YES and we establish separate security by assigning SECADM1=SAMANTHA with SECADM1_TYPE=AUTHID and SECADM2=OFFICER with SECADM2_TYPE=ROLE.

SAMANTHA is an AUTHID with access to TSO and DB2. OFFICER is a ROLE that is available to three people through a trusted context. In Example 3-4 we show the DDL for this scenario.

Example 3-4 Trusted context and ROLE setup

```
CREATE ROLE OFFICER;
CREATE TRUSTED CONTEXT SEC_OFFICER_MARILYN
BASED UPON CONNECTION USING SYSTEM AUTHID MARILYN
ATTRIBUTES (JOBNAME 'MARILYN')
DEFAULT ROLE OFFICER
ENABLE;
CREATE TRUSTED CONTEXT SEC_OFFICER_KATE
BASED UPON CONNECTION USING SYSTEM AUTHID KATE
ATTRIBUTES (JOBNAME 'KATE')
DEFAULT ROLE OFFICER
ENABLE;
CREATE TRUSTED CONTEXT SEC_OFFICER_WILLIAM
BASED UPON CONNECTION USING SYSTEM AUTHID WILLIAM
ATTRIBUTES (JOBNAME 'WILLIAM')
DEFAULT ROLE OFFICER
ENABLE;
```

Example 3-5 shows the setup for the system and database administrators.

Example 3-5 System and database administrators

```
GRANT SYSADM TO PROD_SYSADM;
GRANT DBADM WITH ACCESSCTRL WITH DATAACCESS ON SYSTEM TO PROD_DBA;
```

JOHN and SALLY are connected to RACF group PROD_SYSADM. MARIA is connected to RACF group PROD_DBA. If you want the system database administrators to have access to data but not the ability to grant or revoke privileges, you can grant authorities, as shown in Example 3-6.

Example 3-6 DDL for ACCESSCTRL

```
REVOKE DBADM ON SYSTEM FROM PROD_DBA NOT INCLUDING  DEPENDENT PRIVILEGES;
REVOKE ACCESSCTRL FROM PROD_DBA NOT INCLUDING  DEPENDENT PRIVILEGES;
REVOKE DATAACCESS FROM PROD_DBA NOT INCLUDING  DEPENDENT PRIVILEGES;
GRANT DBADM WITHOUT ACCESSCTRL WITH DATAACCESS ON SYSTEM TO PROD_DBA;
```

3.7.4 Separating system database administration without access control

This scenario describes a high separation of duties. We show how the SYSTEM DBADM authority can be set up to manage all objects across the DB2 subsystem, without the ability to access the data or grant / revoke privileges on the DB2 objects. Different authids are used to control access to data and access to the objects created by the person holding the SYSTEM DBADM authority. This kind of setup can be used in a production environment, as it reduces the privileges normally available to DBAs.

We set SEPARATE_SECURITY DSNZPARM to YES and we establish separate security by assigning SECADM1=SAMANTHA with SECADM1_TYPE=AUTHID and SECADM2=OFFICER with SECADM2_TYPE=ROLE.

SAMANTHA is an AUTHID with access to TSO and DB2. OFFICER is a ROLE that is available to three people through a trusted context. In Example 3-7, we show the DDL for this scenario.

Example 3-7 DDL for authority separation

```
CREATE ROLE OFFICER;
CREATE TRUSTED CONTEXT SEC_OFFICER_MARILYN
BASED UPON CONNECTION USING SYSTEM AUTHID MARILYN
ATTRIBUTES (JOBNAME 'MARILYN')
DEFAULT ROLE OFFICER
ENABLE;
CREATE TRUSTED CONTEXT SEC_OFFICER_KATE
BASED UPON CONNECTION USING SYSTEM AUTHID KATE
ATTRIBUTES (JOBNAME 'KATE')
DEFAULT ROLE OFFICER
ENABLE;
CREATE TRUSTED CONTEXT SEC_OFFICER_WILLIAM
BASED UPON CONNECTION USING SYSTEM AUTHID WILLIAM
ATTRIBUTES (JOBNAME 'WILLIAM')
DEFAULT ROLE OFFICER
ENABLE;
```

In Example 3-8, we show the setup for the system and database administrators.

Example 3-8 System and database administrators without access

```
GRANT SYSADM TO PROD_SYSADM;
GRANT DBADM WITHOUT ACCESSCTRL WITHOUT DATAACCESS ON SYSTEM TO PROD_DBA;
```

JOHN and SALLY are connected to RACF group PROD_SYSADM. MARIA is connected to RACF group PROD_DBA.

In Example 3-9, we grant the DATAACCESS and ACCESSCTRL authorities to two different RACF groups. One needs access to the data, and the other manages privileges on DB2 objects.

Example 3-9 DCL for data access

```
GRANT DATAACCESS TO PROD_DATA;
GRANT ACCESSCTRL TO PROD_ACCESS;
```

HARRY is connected to RACF group PROD_DATA. ARNOLD is connected to RACF group PROD_ACCESS.

3.7.5 Keeping security simple

The number of people who can be install SYSADM, SYSADM, SECADM, or SYSCTRL should be small. These names should all be groups or roles, and the number of people able to use those groups or roles should be small. A rule of thumb would be 10 people, most of whom do not use this authority for their normal work.

All access with SYSADM or SYSCTRL authority should be audited. Other administrative users should be restricted to the access needed and also be groups or roles. Where the work is sensitive, auditing is required. SYSOPER can be controlled. MONITOR2 should only be provided to those who can view all work on the DBMS. Public access should be avoided without careful justification and understanding of the security policy. The BINDAGENT privilege is relatively weak security. Grant BINDAGENT from an ID with only the needed authority, not SYSADM in general.

3.7.6 Dependent privileges with new authorities

When revoking new authorities such as SYSTEM DBADM, DATAACCESS, and ACCESSCTRL, the NOT INCLUDING DEPENDENT PRIVILEGES clause must be used. No cascading revokes take place and this makes the transition easier when a person holding such an authority changes jobs.

3.8 Revoking without cascade

Removing privileges from an authid can have a negative impact if that authid was granted those privileges with an SQL GRANT statement that had the WITH GRANT option.

Assume authid DB2R2 is granted a privilege by authid DB2R1. DB2R1 had been granted that privilege from DB2R3 WITH GRANT OPTION. When DB2R3 revokes the privilege from DB2R1, the revoke cascades and authid DB2R2 also loses that privilege. That process is known as a cascade revoke. This might be an unwanted effect.

Here is another example: You might need to revoke a SYSADM authority, but you do not want to lose privileges that are granted by that authority.

3.8.1 Controlling cascade

DB2 10 for z/OS has a new DSNZPARM and a new SQL REVOKE clause to control the cascade:

- REVOKE_DEP_PRIVILEGES

  The new system parameter REVOKE_DEP_PRIVILEGES on the DSN6SPRM macro, or REVOKE DEP PRIV field on the installation panel DSNTIPP1, gives you the choice of three options:

NO	Prevents removal of dependent privileges or prevents a cascade revoke.
YES	Allows the removal of dependent privileges, allowing a cascade revoke.
SQLSTMT	Lets you decide whether or not removing dependent privileges, or cascading the REVOKE, should be allowed by specifying a new clause on the SQL REVOKE statement. This is the default.

  The system parameter REVOKE_DEP_PRIVILEGES is updatable using the DB2 –SET SYSPARM command.

- SQL REVOKE

  The SQL REVOKE has two new clauses to support whether revocation of privileges from an authid removes the privileges granted by that authid:

  – INCLUDING DEPENDENT PRIVILEGES

    The old behavior. When an authorization or privilege is revoked from an authid or a role, any dependent authorizations or privileges are also revoked. This clause cannot be specified if the subsystem parameter REVOKE_DEP_PRIVILEGES has been set to NO.

  – NOT INCLUDING DEPENDENT PRIVILEGES

    The new option. With this clause, cascade revoke is prevented. This clause cannot be specified if the subsystem parameter REVOKE_DEP_PRIVILEGES has been set to YES.

  The default for the SQL REVOKE statement is INCLUDING DEPENDENT PRIVILEGES unless one of these two conditions exists:

  – If ACCESSCTRL, DATAACCESS or SYSTEM DBADM is being revoked, there is no cascade. The default is always NOT INCLUDING DEPENDENT PRIVILEGES and must be specified on the REVOKE SQL statement.

  – When the subsystem parameter REVOKE_DEP_PRIVILEGES is set to NO, the default is always NOT INCLUDING DEPENDENT PRIVILEGES.

3.8.2 Removing DBA privileges to improve auditing

We describe in this scenario a production environment where the DBAs currently have the CREATEDBA privilege and we want to remove this privilege and change to a model with SYSTEM DBADM. The intention is also to simplify the auditing of the production activities by implementing a single audit policy. We show how this policy is in effect whenever DB2 starts. This policy can only be stopped by an authid holding the SECADM authority.

Current setup

DB2R2 holds the SYSADM authority, and SEPARATE_SECURITY is set to NO.

In Example 3-10, we show how DB2R2 grants privileges to RACF group PRODDBA.

Example 3-10 Privileges granted to PRODDBA

```
GRANT USE OF BUFFERPOOL BP2 TO PRODDBA;
GRANT USE OF BUFFERPOOL BP3 TO PRODDBA;
GRANT CREATEDBA TO PRODDBA;
```

The production DBAs connect to the PRODDBA RACDF group:

```
CO (PAOLOR8 PAOLOR9) GROUP(PRODDBA)
```

In Example 3-11, we show how PAOLOR9 uses the CREATEDBA privilege to create a database. The RACF group PRODDBA now has the DBADM privilege over database DMTGS.

Example 3-11 Creation of database

```
SET CURRENT SQLID='PRODDBA';
...
CREATE DATABASE DMTGS
...
CREATE TABLESPACE SMTGABC2 IN DMTGS
...
CREATE TABLE MTG.MORTGAGES
...
```

In Example 3-12, we show how PAOLOR9 now grants privileges to business users and to production operations people.

Example 3-12 Grant privileges to business users

```
SET CURRENT SQLID='PRODDBA';
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+--------
GRANT SELECT ON MTG.MORTGAGES TO MTGUSER;
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+--------
GRANT DISPLAYDB,IMAGCOPY,LOAD,RECOVERDB,REORG,STARTDB,STATS,STOPDB,
      REPAIR ON DATABASE DMTGS TO PRODOPS;
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+--------
```

PAOLOR7 is a business user connected to RACF group MTGUSER, In Example 3-13, we show that he is able to read the mortgage data.

Example 3-13 Accessing the MORTGAGES table

```
SET CURRENT SQLID='MTGUSER';
---------+---------+---------+---------+---------+---------+----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+----
```

```
SELECT * FROM MTG.MORTGAGES;
---------+---------+---------+---------+---------+---------+----
       REF       LOAN_AMT  MAT          TERM  INTRATE  PROPTYPE
---------+---------+---------+---------+---------+---------+----
     1111.      500000.00  2020-01-31    34.    6.500        1.
     2222.      200000.00  2020-03-31    24.    4.440        1.
     3333.      250000.00  2021-01-31    12.    5.440        1.
...
```

PAOLOR6 is in the operations team and is connected to RACF group PRODOPS. PAOLOR6 can make the database temporarily inaccessible for maintenance. In Example 3-14, we show how PAOLOR7 exercises that privilege.

Example 3-14 Stopping the database

```
Cmd 1 ===> -STOP DB(DMTGS)
...
DSN9022I  -DBOC DSNTDDIS 'STOP DATABASE' NORMAL COMPLETION
***
```

New approach

For the new approach, we use revoke without cascade, which is done by using the clause:

```
NOT INCLUDING DEPENDENT PRIVILEGES
```

We use the new SYSTEM DBADM authority and revoke the CREATEDBA privilege. We also implement an audit policy to continually monitor the use of the SYSTEM DBADM authority.

First, DB2R2, holding SYSADM authority, implements an audit policy that will always monitor the use of the SYSTEM DBADM authority. In Example 3-15, we show how a row is inserted into the audit policies table. This policy will be in effect at DB2 start and can only be stopped by an authid holding the SECADM authority. This policy indicates that auditing is enabled for when the SYSTEM DBADM authority is used to perform database administration tasks.

Example 3-15 Auditing SYSTEM DBADM

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES(AUDITPOLICYNAME, DBADMIN ,DB2START)
 VALUES( 'SYS_DBADM_POL', 'B', 'S');
---------+---------+---------+---------+---------+---------+---------+--
DSNE615I NUMBER OF ROWS AFFECTED IS 1
```

DB2R2 starts the audit policy trace, as shown in Example 3-16.

Example 3-16 Start the trace

```
-STA TRACE (AUDIT) DEST (SMF) AUDTPLCY(SYS_DBADM_POL)
....
DSNW130I  -DBOC AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
 DSNW192I  -DBOC AUDIT POLICY SUMMARY
 AUDIT POLICY SYS_DBADM_POL STARTED
 END AUDIT POLICY SUMMARY
 DSN9022I  -DBOC DSNWVCM1 '-STA TRACE' NORMAL COMPLETION
 ***
```

In Example 3-17, DB2R2 grants the new SYSTEM DBADM authority to the existing RACF PRODDBA group.

Example 3-17 Grant DBADM on SYSTEM

```
GRANT DBADM WITHOUT DATAACCESS WITHOUT ACCESSCTRL ON SYSTEM TO PRODDBA;
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

Attention: When an authid (connected to group PRODDBA) creates a database, that authid will have access to data in that database and will be able to grant privileges on objects in that database. A table created using a different authority (for example, SYSADM, CREATEDBA) will not be accessible (-514) to the production DBA PAOLOR9 (connected to PRODDBA).

in Example 3-18, we show how DB2R2 now revokes the CREATEDBA privilege without affecting what has been granted by the production DBAs.

Example 3-18 Revoke without cascade

```
REVOKE CREATEDBA FROM PRODDBA NOT INCLUDING DEPENDENT PRIVILEGES;
---------+---------+---------+---------+---------+---------+------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

In Example 3-19, we show how PAOLOR6, the operations person, can still issue commands against the database.

Example 3-19 Start the database

```
Cmd 1 ===> -START DB(DMTGS)
...
DSN9022I  -DBOC DSNTDDIS 'START DATABASE' NORMAL COMPLETION
***
```

In Example 3-20, we show how PAOLOR9, the production DBA, can still grant privileges on the mortgage table. A new RACF group OFFLINE has been set up just for the purpose of running the nightly offline maintenance window.

Example 3-20 Grant privilege to new group

```
SET CURRENT SQLID='PRODDBA';
---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+--------
GRANT SELECT ON TABLE MTG.MORTGAGES TO OFFLINE;
---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

In Example 3-21, we show how the business user PAOLOR7 still has the ability to read the mortgage table.

Example 3-21 Access the MORTGAGES table

```
---------+---------+---------+---------+---------+---------+------
SET CURRENT SQLID='MTGUSER';
---------+---------+---------+---------+---------+---------+------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

```
---------+---------+---------+---------+---------+---------+------
SELECT * FROM MTG.MORTGAGES;
---------+---------+---------+---------+---------+---------+------
       REF       LOAN_AMT  MAT          TERM  INTRATE  PROPTYPE
---------+---------+---------+---------+---------+---------+------
     1111.      500000.00  2020-01-31    34.    6.500        1.
     2222.      200000.00  2020-03-31    24.    4.440        1.
     3333.      250000.00  2021-01-31    12.    5.440        1.
     4444.       80000.00  2022-01-31    24.    4.330        1.
```

Conclusion

We have shown how easy it is to revoke without cascade and in this example simplify the auditing of the production DBAs.

We could have further secured the production DBAs access, for example, we could have granted the SYSTEM DBADM authority to a role and only made that role available to a DBA in a local trusted context through TSO.

3.9 Debugging and performance analysis privileges

In this section, we examine the new EXPLAIN privilege and the SQLADM authority.

3.9.1 EXPLAIN privilege

The EXPLAIN statement obtains information about access path selection for an explainable statement. A statement is explainable if it is a SELECT or INSERT statement, or the searched form of an UPDATE or DELETE statement. The information obtained is placed in a user-supplied plan table. EXPLAIN is a powerful tool that provides information about how a particular SQL statement will be processed by the DB2 optimizer. SQL coding best practices for performance always recommend performing predictive analysis of an SQL statement performance prior to implementing into production. Many customers perform access path regression analysis whenever migrating a new version of an application into production, by using EXPLAIN, to prevent unexpected application performance problems.

Historically, the use of EXPLAIN required that the authorization identifier who was issuing the EXPLAIN statement had to have DB2 authorization to the statement being explained. In practical terms, for an INSERT SQL statement to be inserted into a production table which was in a bound application, this meant that the ID issuing the EXPLAIN needed to be granted INSERT privileges to that table. Obviously, this situation presented significant challenges from a security perspective, and we see in many customer implementations, a high level of access authority is being granted to application developers and performance analysis specialists, for no other purpose than to allow for the use of EXPLAIN in the course of problem determination.

Development DBAs and programmers typically have had to ask someone in the production environment to provide the current plan table content for static SQL or request a current explain to be run.

With DB2 10 for z/OS, we now see several new authorization and privileges types that can allow customers to better manage the activities associated with SQL performance management tasks, such as EXPLAIN.

The EXPLAIN system privilege provides you the ability to issue EXPLAIN, PREPARE, and DESCRIBE statements without requiring the privilege to execute the statements. It allows you to validate your applications and SQL syntaxes, without requiring access to data, before putting them into production. With the EXPLAIN authority, you are able to provide the ability to conduct performance analysis activity to developers and performance analysts, without having to provide them data access privileges. The EXPLAIN privilege also extends authorization to DESCRIBE and PREPARE statements as well.

The ability to use EXPLAIN without having access to the data is an enhancement that speeds up investigation of problems. A second big benefit is that security is improved, as performance experts do not require access to the data and can now use the minimum privileges to do their job

Prior to DB2 10 for z/OS, many customers were unwilling to extend the necessary table privileges for EXPLAIN processing to developers. As a result, many customers were unable to conduct the level of predictive performance analysis prior to migrating new statements into production, which resulted in application performance degradation, application availability disruptions, and additional development and support cycles needed to correct poorly performing SQL. With the new DB2 10 EXPLAIN authority, we expect that customers will now be able to widely distribute the capability to EXPLAIN to all of their application developers, encouraging more performance inspection and analysis, without any increased exposure to application data.

The EXPLAIN system privilege has the following capabilities:

- The ability to issue an SQL EXPLAIN ALL statement without having the privileges to execute that SQL statement.
- The ability to issue SQL PREPARE and DESCRIBE TABLE statements without requiring any privileges on the object.
- The ability to specify new BIND EXPLAIN(ONLY) and SQLERROR(CHECK) options.
- The ability to execute EXPLAIN dynamic SQL statements under a new special register called CURRENT EXPLAIN MODE = EXPLAIN.

Consider making this privilege available to all developers through a secondary authid or a ROLE. The following sample GRANT statement enables developers to use EXPLAIN on all the SQL in the subsystem:

```
GRANT EXPLAIN ON SYSTEM TO PGMRS;
```

3.9.2 SQLADM

One other operational issue is that many popular system level and SQL performance monitoring products have the capabilities to collect SQL statements that are currently executing and to be able to launch EXPLAIN dynamically. To take advantage of these features, many customers create authorization identifiers for use with these products, which have high level of system authorities.

DB2 10 for z/OS introduces a new system authority called SQLADM. The SQLADM authority allows you to issue SQL EXPLAIN statements, execute PROFILE commands, run the RUNSTATS and MODIFY STATISTICS utilities on all user databases, and execute system-defined routines, such as stored procedures or functions, and any packages that are executed within the routines.

Only an authorization ID or a role with the SECADM or ACCESSCTRL authority can grant or revoke the SQLADM authority. With the SQLADM authority, you have implicit SELECT access on all the catalog tables and implicit INSERT, DELETE, and UPDATE privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES).

The SQLADM authority contains the following privileges:

- EXPLAIN
- MONITOR1
- MONITOR2
- Select on the catalog tables
- RUNSTATS
- MODIFY STATISTICS
- EXECUTE on system defined routines and functions

The SQLADM authority cannot access data or perform DDL.

SQLADM has no other system authorities, and in general, the scope of the SQLADM authority includes all of the privileges that the typical performance analyst would need to perform their duties. In addition, the SQLADM authority contains the necessary privileges that would be needed for many of the popular performance management products to use, which could reduce the need for these products to be associated with authorization identifiers with higher levels of privilege, such as SYSADM.

Important: Although SQLADM provides flexibility for separating performance analysis tasks from system administration authority, and reducing the requirement for SYSADM, remember that with MONITOR2, all of the IFI trace classes and IFCIDs can be collected. When used inappropriately, MONITOR2 might expose sensitive information as part of the IFI event collection process, so the use of SQLADM authority should be carefully controlled.

We show a sample GRANT statement that enables PAOLOR9 (connected to RACF group PROD_PERF) to administer all the SQL in the SYSTEM without having access to the data:

```
GRANT SQLADM ON SYSTEM TO PROD_PERF;
```

3.10 DSNZPARMs related to security

Several DB2 system parameters have a direct or indirect impact on security. In this section, we provide a brief description of these system parameters by referencing detailed documentation.

3.10.1 Installation panels

There are two installation panels devoted to security: DSNTIPP and DSNTIPP1. DSNTIPR also has a security relater parameter for DDF. For more details, refer to *DB2 10 for z/OS Installation and Migration Guide,* GC19-2974.

Protection panel: DSNTIPP

The entries on the DSNTIPP panel are all related to security matters (Figure 3-2).

```
DSNTIPP INSTALL DB2 - PROTECTION
===> _
Enter data below:
1 ARCHIVE LOG RACF   ===> NO RACF protect archive log data sets
2 USE PROTECTION     ===> YES DB2 authorization enabled. YES or NO
3 PLAN AUTH CACHE    ===> 3072 Size in bytes per plan (0 - 4096)
4 PACKAGE AUTH CACHE ===> 5M Global - size in bytes (0-10M)
5 ROUTINE AUTH CACHE ===> 5M Global - size in bytes (0-10M)
6 AUTH EXIT LIMIT    ===> 10 Access control exit shutdown threshold
PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 3-2 Protection panel - DSNTIPP

In the following list, for each entry, we first mention the subsystem parameter, the field that is specified in the panel, and then a brief description, following the sequence in the panel:

- PROTECT (ARCHIVE LOG RACF field)

  The PROTECT subsystem parameter specifies whether archive log data sets are to be protected with *individual profiles* with RACF when they are created.

  If you specify YES, RACF protection must be active for DB2. You need to define RACF individual profiles, not RACF generic profiles for the archive log data sets. In addition, if you archive to tape, the TAPEVOL RACF class must be active.

- AUTH (USE PROTECTION field)

  The AUTH subsystem parameter controls whether DB2 checks authorizations. If it is set to NO, all privileges are granted to PUBLIC, which allows any user to access user data.

- AUTHCACH (PLAN AUTH CACHE field)

  The AUTHCACH subsystem parameter specifies the size (in bytes per plan) of the authorization cache that is to be used if no CACHESIZE is specified on the BIND PLAN subcommand.

- CACHEPAC (PACKAGE AUTH CACHE field)

  The CACHEPAC subsystem parameter determines the amount of storage in bytes to allocate for caching package authorization information for all packages on this DB2 member. The cache is stored in the DSN1DBM1 address space. Changes that are made in RACF profiles are not immediately reflected. This setting might result in a user with cached AUTHID being able to run a package.

- CACHERAC (ROUTINE AUTH CACHE field)

  The CACHERAC subsystem parameter determines the amount of storage in bytes to allocate for caching routine authorization information for all routines on this DB2 member. Routines include stored procedures and user-defined functions. Changes that are made in RACF profiles are not immediately reflected. This setting might result in a user (cached AUTHID) being able to invoke a routine or function.

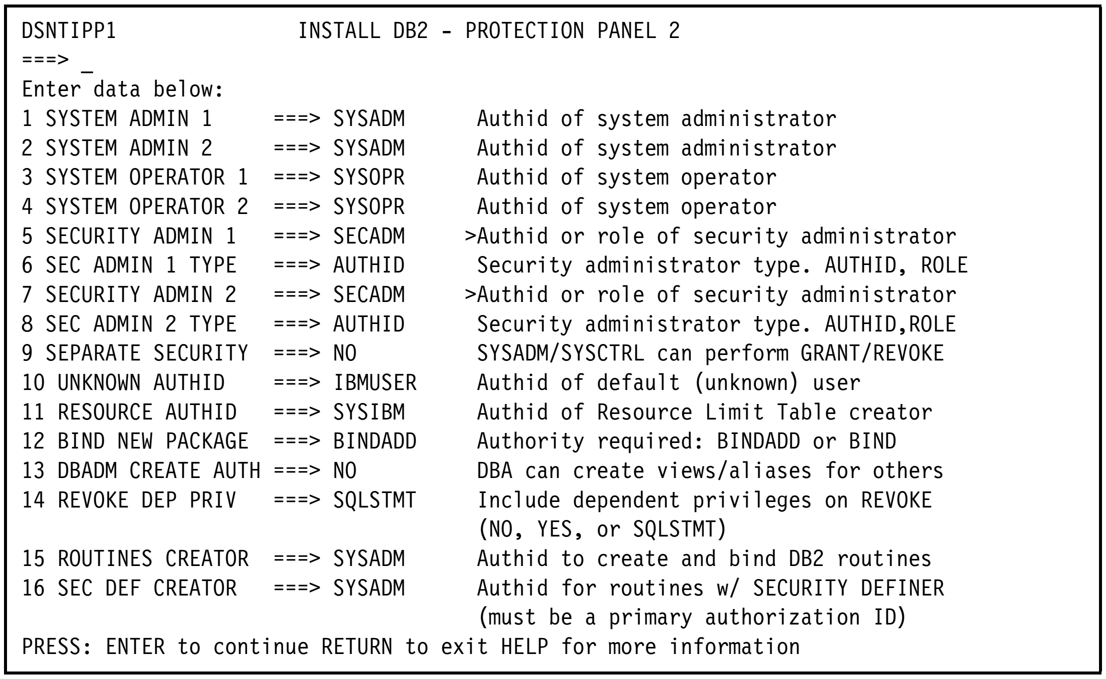
- AEXITLIM (AUTH EXIT LIMIT field)

  The AEXITLIM subsystem parameter controls the number of abends of the DB2 access control authorization exit routine that are to be tolerated before it is shut down. After the exit routine shuts down, it can be reactivated only by restarting DB2.

Protection panel 2: DSNTIPP1

You use the DSNTIPP1 panel to specify security options, including the authorization IDs that are required for system administrators and system operators, and the authorization IDs or roles that are required for security administrators. You can also specify the authorization IDs that are to be used to install, configure, and validate DB2-supplied routines.

Some input fields are defined as ISPF scrollable fields (>), as shown in Figure 3-3.



```
DSNTIPP1               INSTALL DB2 - PROTECTION PANEL 2
===> _
Enter data below:
1 SYSTEM ADMIN 1     ===> SYSADM      Authid of system administrator
2 SYSTEM ADMIN 2     ===> SYSADM      Authid of system administrator
3 SYSTEM OPERATOR 1  ===> SYSOPR      Authid of system operator
4 SYSTEM OPERATOR 2  ===> SYSOPR      Authid of system operator
5 SECURITY ADMIN 1   ===> SECADM     >Authid or role of security administrator
6 SEC ADMIN 1 TYPE   ===> AUTHID      Security administrator type. AUTHID, ROLE
7 SECURITY ADMIN 2   ===> SECADM     >Authid or role of security administrator
8 SEC ADMIN 2 TYPE   ===> AUTHID      Security administrator type. AUTHID,ROLE
9 SEPARATE SECURITY  ===> NO          SYSADM/SYSCTRL can perform GRANT/REVOKE
10 UNKNOWN AUTHID    ===> IBMUSER     Authid of default (unknown) user
11 RESOURCE AUTHID   ===> SYSIBM      Authid of Resource Limit Table creator
12 BIND NEW PACKAGE  ===> BINDADD     Authority required: BINDADD or BIND
13 DBADM CREATE AUTH ===> NO          DBA can create views/aliases for others
14 REVOKE DEP PRIV   ===> SQLSTMT     Include dependent privileges on REVOKE
                                      (NO, YES, or SQLSTMT)
15 ROUTINES CREATOR  ===> SYSADM      Authid to create and bind DB2 routines
16 SEC DEF CREATOR   ===> SYSADM      Authid for routines w/ SECURITY DEFINER
                                      (must be a primary authorization ID)
PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 3-3 Protection panel 2 - DSNTIPP1

In the following list, for each entry, we first mention first the subsystem parameter, the field that is specified in the panel, and then a brief description, following the sequence in the panel:

- SYSADM (SYSTEM ADMIN 1 field)

  The SYSADM subsystem parameter specifies the first of two authorization IDs that will have installation SYSADM authority.

- SYSADM2 (SYSTEM ADMIN 2 field)

  The authorization ID that is specified for this parameter can manage the DB2 subsystem even when the DB2 catalog is unavailable. This authorization ID can access all user data and can run any application.

- SYSOPR1 (SYSTEM OPERATOR 1 field)

  The authorization ID that is specified for this parameter can issue most of the DB2 commands, such as BIND QUERY, CANCEL THREAD, and STOP DB2. This ability might provide the authorization ID with information about user data. Also, the authorization ID can run the DSN1SDMP utility and terminate any utility job.

- SYSOPR2 (SYSTEM OPERATOR 2 field)

  Like SYSOPR1, the authorization ID that is specified for this parameter can issue most of the DB2 commands, such as BIND QUERY, CANCEL THREAD, and STOP DB2. This ability might provide the authorization ID with information about the user.

- SECADM1 (SECURITY ADMIN 1 field)

  The SECADM1 subsystem parameter specifies the first of two authorization IDs or roles that have the DB2 security administrator authority.

- SECADM1_TYPE (SEC ADMIN 1 TYPE field)

  The value specifies whether the SECADM1 parameter is an authorization ID or a role. Changing the value of this field changes the SECADM2 parameter from an authorization ID to a role, or from a role to an authorization ID.

- SECADM2 (SECURITY ADMIN 2 field)

  The SECADM2 subsystem parameter specifies the second of two authorization IDs or roles that are to have DB2 security administrator authority.

- SECADM2_TYPE (SEC ADMIN 2 TYPE field)

  The value specifies whether the SECADM2 parameter is an authorization ID or a role. Changing the value of this field changes the SECADM2 parameter from an authorization ID to a role, or from a role to an authorization ID.

- SEPARATE_SECURITY (SEPARATE SECURITY field)

  If set to YES, it separates DB2 security administrator duties from system administrator duties. Users with SYSADM authority cannot manage security objects (such as roles and trusted contexts), perform grants, or revoke privileges granted by others. Users with SYSCTRL authority cannot manage roles, perform grants, or revoke privileges granted by others. However, existing grants made by users with SYSADM or SYSCTRL authority are unchanged. SECADM or ACCESSCTRL authority is required for security administration.

  Before setting SEPARATE SECURITY to YES, set at least one SECADM system parameter to an authorization ID, or create the necessary trusted contexts and roles. If you specify YES, system administrator authority can no longer be used to perform security tasks, and the SECADM authority is required to manage security objects such as trusted contexts and roles. If both SECADM system parameters are set to roles and those roles have not been created, no one will have the authority to manage security objects.

- DEFLTID (UNKNOWN AUTHID field)

  The DEFLTID subsystem parameter specifies the authorization ID that is to be used if RACF is not available for batch access and USER= is not specified in the JOB statement.

- RLFAUTH (RESOURCE AUTHID field)

  The RLFAUTH subsystem parameter specifies the authorization ID that is to be used if you plan to use the resource limit facility (governor).

- BINDNV (BIND NEW PACKAGE field)

  When this parameter is set to BIND, the BINDADD privilege check is bypassed, and a user with PACKADM authority can create new packages.

- DBACRVW (DBADM CREATE AUTH field)

  The parameter controls whether an authorization ID or role with DBADM authority on a database is allowed to create views, aliases, and materialized query tables for another authorization ID or role. If it is set to YES, authorization IDs and roles with DBADM authority on a database can give others access to user data.

- ROUTINES CREATOR field

  The value specifies the CURRENT SQLID setting that is used when creating, configuring, and validating most DB2-supplied routines. This field also specifies the default OWNER that is used when binding packages for these routines. It is a parameter of program DSNTRIN in job DSNTIJRT.

- SEC DEF CREATOR field

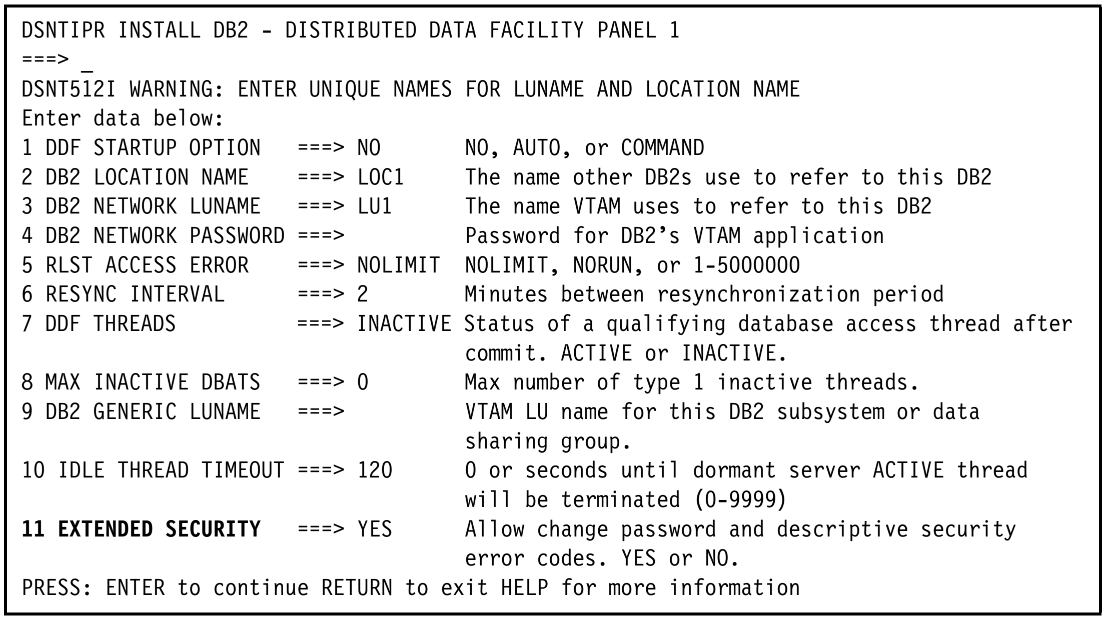
  The value specifies the CURRENT SQLID setting that is to be used when creating and configuring DB2-supplied routines that are defined with the SECURITY DEFINER option. It is a parameter of program DSNTRIN in job DSNTIJRT.

- REVOKE_DEP_PRIVILEGES (REVOKE DEP PRIV field)

  If it is set to NO, privileges that were granted by a user are retained even if that user loses the authority that allowed the user to perform the grant.

Distributed Data Facility panel 1: DSNTIPR

The entries on the DSNTIPR panel shown in Figure 3-4 control the starting of the distributed data facility (DDF).



```
DSNTIPR INSTALL DB2 - DISTRIBUTED DATA FACILITY PANEL 1
===> _
DSNT512I WARNING: ENTER UNIQUE NAMES FOR LUNAME AND LOCATION NAME
Enter data below:
1 DDF STARTUP OPTION   ===> NO       NO, AUTO, or COMMAND
2 DB2 LOCATION NAME    ===> LOC1     The name other DB2s use to refer to this DB2
3 DB2 NETWORK LUNAME   ===> LU1      The name VTAM uses to refer to this DB2
4 DB2 NETWORK PASSWORD ===>          Password for DB2's VTAM application
5 RLST ACCESS ERROR    ===> NOLIMIT  NOLIMIT, NORUN, or 1-5000000
6 RESYNC INTERVAL      ===> 2        Minutes between resynchronization period
7 DDF THREADS          ===> INACTIVE Status of a qualifying database access thread after
                                     commit. ACTIVE or INACTIVE.
8 MAX INACTIVE DBATS   ===> 0        Max number of type 1 inactive threads.
9 DB2 GENERIC LUNAME   ===>          VTAM LU name for this DB2 subsystem or data
                                     sharing group.
10 IDLE THREAD TIMEOUT ===> 120      0 or seconds until dormant server ACTIVE thread
                                     will be terminated (0-9999)
11 EXTENDED SECURITY   ===> YES      Allow change password and descriptive security
                                     error codes. YES or NO.
PRESS: ENTER to continue RETURN to exit HELP for more information
```

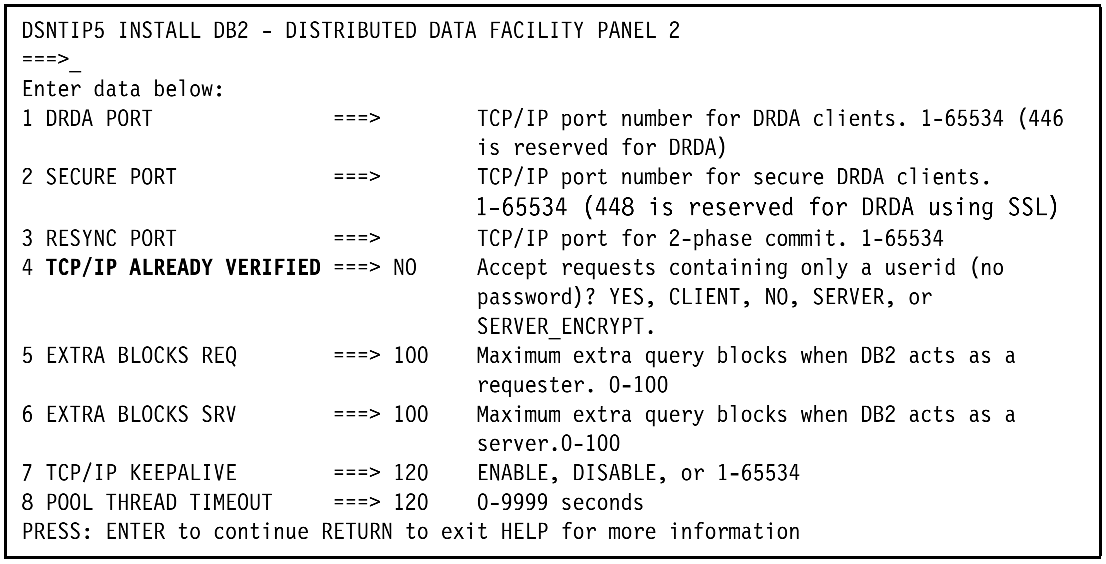
Figure 3-4 Distributed data facility panel - DSNTIPR

EXTSEC (EXTENDED SECURITY field): The EXTSEC subsystem parameter specifies how two related security options are to be set. These settings control what happens when a DDF connection has security errors and whether RACF users can change their passwords through the DRDA change password function.

When this parameter is set to YES, detailed reason codes are returned to the client when a DDF connection request fails because of security errors that might enable more malicious attacks. If this parameter is set to YES, RACF users can change their passwords by using the DRDA change password function.

Distributed data facility panel 2: DSNTIP5

The entries on the DSNTIP5 panel are used to configure the distributed data facility (DDF), as shown in Figure 3-5.



```
DSNTIP5 INSTALL DB2 - DISTRIBUTED DATA FACILITY PANEL 2
===>_
Enter data below:
1 DRDA PORT              ===>       TCP/IP port number for DRDA clients. 1-65534 (446
                                    is reserved for DRDA)
2 SECURE PORT            ===>       TCP/IP port number for secure DRDA clients.
                                    1-65534 (448 is reserved for DRDA using SSL)
3 RESYNC PORT            ===>       TCP/IP port for 2-phase commit. 1-65534
4 TCP/IP ALREADY VERIFIED ===> NO   Accept requests containing only a userid (no
                                    password)? YES, CLIENT, NO, SERVER, or
                                    SERVER_ENCRYPT.
5 EXTRA BLOCKS REQ       ===> 100   Maximum extra query blocks when DB2 acts as a
                                    requester. 0-100
6 EXTRA BLOCKS SRV       ===> 100   Maximum extra query blocks when DB2 acts as a
                                    server.0-100
7 TCP/IP KEEPALIVE       ===> 120   ENABLE, DISABLE, or 1-65534
8 POOL THREAD TIMEOUT    ===> 120   0-9999 seconds
PRESS: ENTER to continue RETURN to exit HELP for more information
```

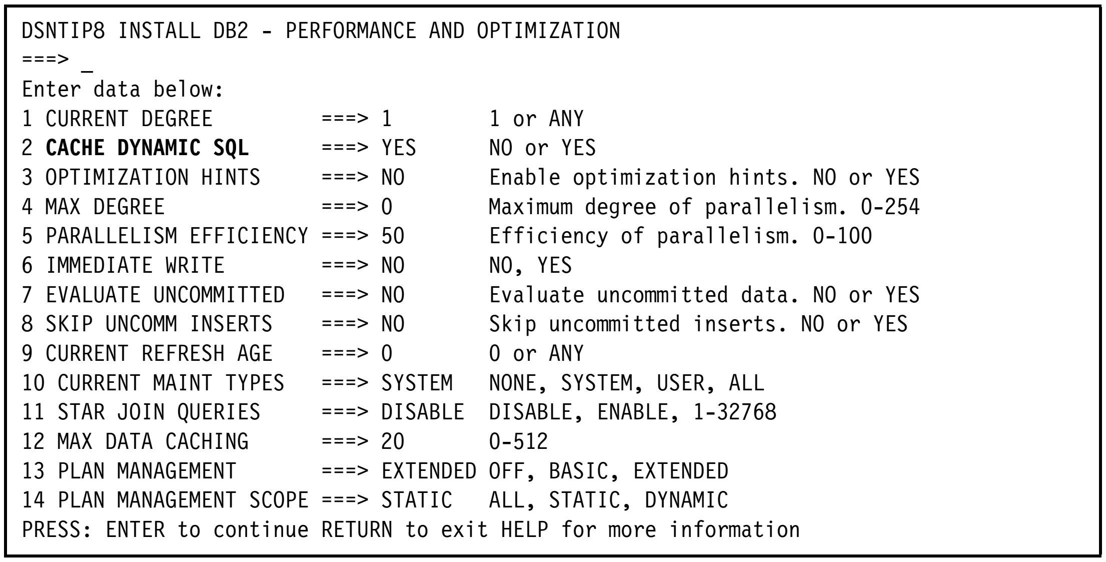
Figure 3-5 Distributed data facility panel - DSNTIP5

TCPALVER (TCP/IP ALREADY VERIFIED field): The TCPALVER subsystem parameter specifies whether DB2 is to accept TCP/IP connection requests that contain only a user ID (no password, RACF PassTicket, or Kerberos ticket). This parameter also specifies if a stronger form of security is required. This parameter is not relevant to trusted context users that have been switched.

Setting the parameter to SERVER_ENCRYPT provides the best security. Connections are accepted only if user credentials are provided to authenticate the user ID, and strong encryption is used to protect the user ID and credentials.

Performance and optimization panel: DSNTIP8

The DSNTIP8 panel (a continuation of the DSNTIP4 panel) sets application programming default values pertaining to performance and optimization, as shown in Figure 3-6.



```
DSNTIP8 INSTALL DB2 - PERFORMANCE AND OPTIMIZATION
===> _
Enter data below:
1 CURRENT DEGREE         ===> 1        1 or ANY
2 CACHE DYNAMIC SQL      ===> YES      NO or YES
3 OPTIMIZATION HINTS     ===> NO       Enable optimization hints. NO or YES
4 MAX DEGREE             ===> 0        Maximum degree of parallelism. 0-254
5 PARALLELISM EFFICIENCY ===> 50       Efficiency of parallelism. 0-100
6 IMMEDIATE WRITE        ===> NO       NO, YES
7 EVALUATE UNCOMMITTED   ===> NO       Evaluate uncommitted data. NO or YES
8 SKIP UNCOMM INSERTS    ===> NO       Skip uncommitted inserts. NO or YES
9 CURRENT REFRESH AGE    ===> 0        0 or ANY
10 CURRENT MAINT TYPES   ===> SYSTEM   NONE, SYSTEM, USER, ALL
11 STAR JOIN QUERIES     ===> DISABLE  DISABLE, ENABLE, 1-32768
12 MAX DATA CACHING      ===> 20       0-512
13 PLAN MANAGEMENT       ===> EXTENDED OFF, BASIC, EXTENDED
14 PLAN MANAGEMENT SCOPE ===> STATIC   ALL, STATIC, DYNAMIC
PRESS: ENTER to continue RETURN to exit HELP for more information
```

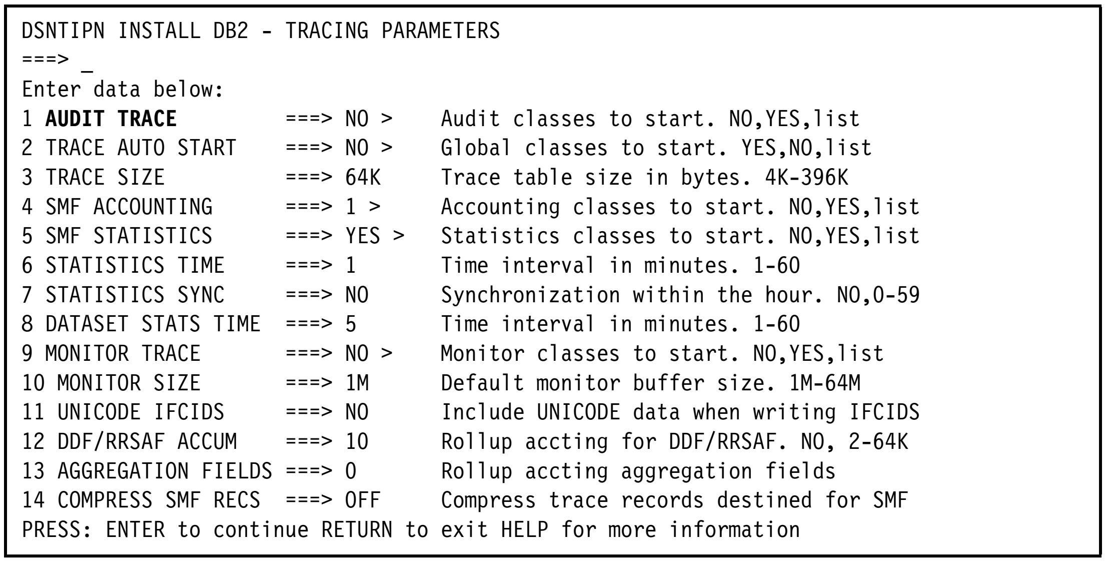
Figure 3-6 Performance and optimization panel - DSNTIP8

CACHEDYN (CACHE DYNAMIC SQL field): The CACHEDYN subsystem parameter controls whether prepared, dynamic SQL statements are to be cached for later use by eligible application processes.

When RACF authorization is used and this parameter is set, the changes that are made in RACF profiles are not immediately reflected. This setting might result in a user (cached AUTHID) being able to issue the dynamic statements until the cache is refreshed.

Tracing parameters panel: DSNTIPN

The entries on the DSNTIPN panel in Figure 3-7 affect the audit, global, accounting, and monitor traces and the checkpoint frequency.



```
DSNTIPN INSTALL DB2 - TRACING PARAMETERS
===> _
Enter data below:
1 AUDIT TRACE          ===> NO >    Audit classes to start. NO,YES,list
2 TRACE AUTO START     ===> NO >    Global classes to start. YES,NO,list
3 TRACE SIZE           ===> 64K     Trace table size in bytes. 4K-396K
4 SMF ACCOUNTING       ===> 1 >     Accounting classes to start. NO,YES,list
5 SMF STATISTICS       ===> YES >   Statistics classes to start. NO,YES,list
6 STATISTICS TIME      ===> 1       Time interval in minutes. 1-60
7 STATISTICS SYNC      ===> NO      Synchronization within the hour. NO,0-59
8 DATASET STATS TIME   ===> 5       Time interval in minutes. 1-60
9 MONITOR TRACE        ===> NO >    Monitor classes to start. NO,YES,list
10 MONITOR SIZE        ===> 1M      Default monitor buffer size. 1M-64M
11 UNICODE IFCIDS      ===> NO      Include UNICODE data when writing IFCIDS
12 DDF/RRSAF ACCUM     ===> 10      Rollup accting for DDF/RRSAF. NO, 2-64K
13 AGGREGATION FIELDS  ===> 0       Rollup accting aggregation fields
14 COMPRESS SMF RECS   ===> OFF     Compress trace records destined for SMF
PRESS: ENTER to continue RETURN to exit HELP for more information
```

Figure 3-7 Tracing panel - DSNTIPN

In the AUDITST (AUDIT TRACE field), the AUDITST parameter controls whether the audit trace is to start automatically when DB2 is started. You can specify the classes for which the audit trace is to start. Audit traces should be enabled to collect information about DB2 security controls to ensure that data access is allowed only for authorized users.

3.10.2 SECADM1 and SECADM2 and role name: Additional details

The DSNZPARMs shown in Table 3-2 are used to configure and manage the SECADM authority.

Table 3-2 DSNZPARMs for SECADM

DSNZPARM	Description
SECADM1	Authorization ID or role name of first SECADM. ► AUTHID: One to eight characters, starting with an alphabetic character. ► ROLE: An ordinary SQL identifier with up to 128 bytes that must not use any of the reserved words ACCESSCTRL, DATAACCESS, DBADM, DBCTRL, DBMAINT, NONE, NULL, PACKADM, PUBLIC, SECADM, or SQLADM. ► Default (if not provided in DSNZPARM): SECADM.
SECADM1_TYPE	Specifies whether SECADM1 is an authorizat.ion ID or a role. ► Acceptable values: AUTHID, ROLE ► Default: AUTHID.
SECADM2	Authorization ID or role name of second SECADM. ► AUTHID: One to eight characters, starting with an alphabetic character. ► ROLE: An ordinary SQL identifier with up to 128 bytes that must not use any of the reserved words ACCESSCTRL, DATAACCESS, DBADM, DBCTRL, DBMAINT, NONE, NULL, PACKADM, PUBLIC, SECADM, or SQLADM. ► Default (if not provided in DSNZPARM): SECADM
SECADM2_TYPE	Specifies whether SECADM2 is an authorization ID or a role. ► Acceptable values: AUTHID, ROLE. ► Default: AUTHID.
SEPARATE_SECURITY	Security administrator duties (SECADM) to be separated from system administrator duties (SYSADM). ► Acceptable values: YES or NO. ► Default value: NO.

The DSNZPARMs described in Table 3-2 are defined on installation panels and are online changeable by either install SYSADM or the SECADM authority. Any attempt by an authorization ID or role to change any of these online DSNZPARMs fails.

The following parameters are not in the installation panels and can be set in the DSN6SPRM macro to allow hexadecimal strings in the SECADM 1 and 2 name definition as ROLEs.

SECADM1_INPUT_STYLE in macro DSN6SPRM

Specifies whether the SECADM1 setting is passed as a hexadecimal string or as a standard character string. Valid values are HEX or CHAR. A value of HEX means that SECADM1 is a hexadecimal character string that represents a Unicode-encoded role name. A value of HEX is valid only when SECADM1_TYPE=ROLE. When SECADM1_INPUT_STYLE is set to HEX, SECADM1 must be an even number of bytes, 2 - 256, consisting entirely of 0 - 9 and A - F.

The default value for SECADM1_INPUT_STYLE is CHAR, which means that SECADM1 is passed as a standard character string. If | SECADM1_TYPE=AUTHID, SECADM1 can be an authorization ID of one to eight characters, or if SECADM1_TYPE=ROLE, SECADM1 can be an ordinary SQL identifier of 1 - 128 characters.

SECADM2_INPUT_STYLE in macro DSN6SPRM

Specifies whether the SECADM2 setting is passed as a hexadecimal string or as a standard character string. Valid values are HEX or CHAR. A value of HEX means that SECADM2 is a hexadecimal character string that represents a Unicode-encoded role name. A value of HEX is valid only when SECADM2_TYPE=ROLE. When SECADM2_INPUT_STYLE is set to HEX, SECADM2 must be an even number of bytes, 2 - 256, consisting entirely of 0 - 9 and A - F.

The default value for SECADM2_INPUT_STYLE is CHAR, which means that SECADM2 is passed as a standard character string. If SECADM2_TYPE=AUTHID, SECADM2 can be an authorization ID of one to eight characters, or if SECADM2_TYPE=ROLE, SECADM2 can be an ordinary SQL identifier of 1 - 128 characters.

3.10.3 Online-updatable subsystem parameters

In previous versions of DB2, users with SYSOPR or higher authorities can update the online-updatable subsystem parameters. In DB2 10, the security-related subsystem parameters can only be updated by users that have SECADM or installation SYSADM authority. The subsystem parameters that are affected by this change are:

- AUTHCACH
- BINDNV
- DBACRVW
- EXTSEC
- TCPALVER

3.10.4 Summary of subsystem parameters related to security

Table 3-3 summarizes the current list of security related parameters.

Table 3-3 DB2 security related parameters

DSNZPARM	Field	Panel or macro	Description
AEXITLIM	AUTH EXIT LIMIT	DSNTIPP	The AEXITLIM subsystem parameter controls the number of abends of the DB2 access control authorization exit routine that are to be tolerated before it is shut down. After the exit routine shuts down, it can be reactivated only by restarting DB2.
AUDITST	AUDIT TRACE	DSNTIPN	Audit traces should be enabled to collect information about DB2 security controls to ensure that data access is allowed only for authorized users.
AUTH	USE PROTECTION	DSNTIPP	If it is set to NO, individual grants cannot be performed and all privileges are granted to PUBLIC, which allows any user to access user data.

DSNZPARM	Field	Panel or macro	Description
AUTHCACH	PLAN AUTH CACHE	DSNTIPB	When RACF authorization is used and this parameter is set, the changes that are made in RACF profiles are not immediately reflected. This setting might result in a user (cached AUTHID) being able to run the application until the cache is refreshed.
BINDNV	BIND NEW PACKAGE	DSNTIPP1	When this parameter is set to BIND, the BINDADD privilege check is bypassed, and a user with PACKADM authority can create new packages.
CACHEDYN	CACHE DYNAMIC SQL	DSNTIP8	When RACF authorization is used (AUTH=YES) and this parameter is set, the changes that are made in RACF profiles are not immediately reflected. This setting might result in a user (cached AUTHID) being able to issue the dynamic statements until the cache is refreshed.
CACHEPAC	PACKAGE AUTH CACHE	DSNTIPP	When RACF authorization is used and this parameter is set, the changes that are made in RACF profiles are not immediately reflected. This setting might result in a user (cached AUTHID) being able to run a package.
CACHERAC	ROUTINE AUTH CACHE	DSNTIPP	When RACF authorization is used and this parameter is set, the changes that are made in RACF profiles are not immediately reflected. This setting might result in a user (cached AUTHID) being able to invoke a routine or function.
DBACRVW	DBADM CREATE AUTH	DSNTIPB	It controls whether an authorization ID or role with DBADM authority on a database is allowed to create views, aliases, and materialized query tables for another authorization ID or role. If it is set to YES, authorization IDs and roles with DBADM authority on a database can give others access to user data.
DEFLTID	UNKNOWN AUTHID	DSNTIPB	The DEFLTID subsystem parameter specifies the authorization ID that is to be used if RACF is not available for batch access and USER= is not specified in the JOB statement.
EXTSEC	EXTENDED SECURITY	DSNTIPB	When this parameter is set to YES, detailed reason codes are returned to the client when a DDF connection request fails because of security errors that might enable more malicious attacks. If this parameter is set to YES, RACF users can change their passwords by using the DRDA change password function.
PRIVATE_ PROTOCOL		DSN6FAC	This parameter is used in DB2 8 and 9 to specify whether private protocol support is enabled (YES) or disabled (NO). PM37300 adds the new option AUTH, Refer to 7.1.4, "Using private protocol authorization" on page 130 for more information. DB2 10 only allows NO or AUTH.

DSNZPARM	Field	Panel or macro	Description
REVOKE_DEP_ PRIVILEGES	REVOKE DEP PRIV	DSNTIPB	If this parameter is set to NO, privileges that were granted by a user are retained even if that user loses the authority that allowed the user to perform the grant.
RLFAUTH	RESOURCE AUTHID	DSNTIPP1	The RLFAUTH subsystem parameter specifies the authorization ID that is to be used if you plan to use the resource limit facility (governor).
None[a]	ROUTINES CREATOR	DSNTIPB	The ROUTINES CREATOR field specifies the CURRENT SQLID setting that is to be used when creating, configuring, and validating most DB2-supplied routines. This field also specifies the default OWNER that is to be used when binding packages for these routines.
SECADM1	SECURITY ADMIN 1	DSNTIPB	The SECADM1 subsystem parameter specifies the first of two authorization IDs or roles that are to have DB2 security administrator authority.
SECADM1_INPUT _STYLE		DSN6SPRM	Specifies whether the SECADM1 setting is passed as a hexadecimal string or a standard character string.
SECADM1_TYPE	SEC ADMIN 1 TYPE	DSNTIPB	The value specifies whether the SECADM1 parameter is an authorization ID or a role. Changing the value of this field changes the SECADM1 parameter from an authorization ID to a role, or from a role to an authorization ID.
SECADM2	SECURITY ADMIN 2	DSNTIPB	The SECADM2 subsystem parameter specifies the second of two authorization IDs or roles that have DB2 security administrator authority.
SECADM2_INPUT _STYLE		DSN6SPRM	Specifies whether the SECADM 2 setting is passed as a hexadecimal string or as a standard character string.
SECADM2_TYPE	SEC ADMIN 2 TYPE	DSNTIPB	The value specifies whether the SECADM2 parameter is an authorization ID or a role. Changing the value of this field changes the SECADM2 parameter from an authorization ID to a role, or from a role to an authorization ID.
None[b]	SEC DEF CREATOR	DSNTIPB	The value of the SEC DEF CREATOR field specifies the CURRENT SQLID setting that is to be used when creating and configuring DB2-supplied routines that are defined with the SECURITY DEFINER option.

DSNZPARM	Field	Panel or macro	Description
SEPARATE_ SECURITY	SEPARATE SECURITY	DSNTIPB	If set to YES, this parameter separates DB2 security administrator duties from system administrator duties. Users with SYSADM authority cannot manage security objects (such as roles and trusted contexts), perform grants, or revoke privileges granted by others. Users with SYSCTRL authority cannot manage roles, perform grants, or revoke privileges granted by others. However, existing grants made by users with SYSADM or SYSCTRL authority are unchanged. SECADM or ACCESSCTRL authority is required for security administration. Before setting SEPARATE SECURITY to YES, set at least one SECADM system parameter to an authorization ID, or create the necessary trusted contexts and roles. If you specify YES, the system administrator authority can no longer be used to perform security tasks, and the SECADM authority is required to manage security objects, such as trusted contexts and roles. If both SECADM system parameters are set to roles and those roles have not been created, no one will have the authority to manage security objects.
SYSADM	SYSTEM ADMIN 1	DSNTIPP	The SYSADM subsystem parameter specifies the first of two authorization IDs that have installation SYSADM authority.
SYSADM2	SYSTEM ADMIN 2	DSNTIPP	The authorization ID that is specified for this parameter can manage the DB2 subsystem even when the DB2 catalog is unavailable. This authorization ID can access all user data and can run any application.
SYSOPR1	SYSTEM OPERATOR 1	DSNTIPB	The authorization ID that is specified for this parameter can issue most of the DB2 commands, such as BIND QUERY, CANCEL THREAD, and STOP DB2. This ability might provide the authorization ID with information about user data. Also, the authorization ID can run the DSN1SDMP utility and terminate any utility job.
SYSOPR2	SYSTEM OPERATOR 2	DSNTIPB	Like SYSOPR1, the authorization ID that is specified for this parameter can issue most of the DB2 commands, such as BIND QUERY, CANCEL THREAD, and STOP DB2. This ability might provide the authorization ID with information about the user.

DSNZPARM	Field	Panel or macro	Description
TCPALVER	TCP/IP ALREADY VERIFIED	DSNTIPB	If the parameter is set to YES or CLIENT, connections are accepted with a user ID only. Security credentials such as a password are not required to authenticate the user ID that is associated with the connection. Setting the parameter to SERVER_ENCRYPT provides the best security. Connections are accepted only if user credentials are provided to authenticate the user ID, and strong encryption is used to protect the user ID and credentials.

a. The value is used to customize the AUTHID parameter of program DSNTRIN in job DSNTIJRT.

b. The value is used to customize the SECDEFID parameter of program DSNTRIN in job DSNTIJRT.

4

Roles and trusted contexts

In this chapter, we provide a high level description of roles and trusted contexts and how these objects are now considered to be security-related in DB2 10.

Roles and trusted contexts were introduced in DB2 9, and there are no changes in DB2 10 to these database entities.

However, in DB2 10, the following objects are now considered to be security-related:

- Row permission
- Column mask
- Role
- Trusted context

Also in DB2 10, with the SEPARATE_SECURITY DSNZPARM set to YES, only security administrators holding the SECADM authority can manage security-related objects in DB2.

If you currently have an environment with many people holding SYSADM authority, removing the privilege to create trusted contexts/roles from SYSADM authority can be a consideration in moving to an environment with SEPARATE_SECURITY set to yes. You can separate the management of security-related object from the system administrators.

In this chapter, we provide some basic information and examples using the new administrative authorities in DB2 10.

These are the two database entities:

- Trusted context
- Role

A trusted context establishes a trusted relationship between DB2 and an external entity, such as a middleware server or another DB2 subsystem. At connection time, a series of trust attributes are evaluated to determine if a specific context can be trusted. After a trusted connection is established, it is possible to acquire, through a role, a special set of privileges for a DB2 authorization ID within the specific connection that are not available to it outside of the trusted connection.

When defined, connections from specific users through defined attachments and source servers allow trusted connections to DB2. The users in this context can also be defined to obtain a database role.

Users must be given permission to use a trusted context. A trusted context can exist without a role. A role is usable only within an established trusted connection based on a predefined trusted context.

This chapter contains the following topics:

- Existing challenges
- Roles
- Trusted contexts
- Challenges addressed by roles and trusted contexts
- Example of a local trusted context: Securing DBA activities
- Example of a remote trusted connection
- Example of a remote trusted connection with multiple users
- Protecting new DB2 10 administrative authorities

4.1 Existing challenges

In this section, we provide a summary of existing challenges for providing the trusted context and role capabilities. For more details about these specific examples, see *Securing DB2 and Implementing MLS on z/OS*, SG24-6480.

4.1.1 Trusting all connection requests

DB2 receives requests from many external entities. Currently, you have the option to set a system parameter that indicates to DB2 that all connections are to be trusted. Trusted means that users do not have to be authenticated to RACF with credentials. It is unlikely that all connection types, such as DRDA, RRS, TSO and batch, from all sources will fit into this category.

4.1.2 Application server user ID / password (three-tier architecture)

Most existing application servers connect to DB2 using a single user ID to initiate requests to DB2. This ID must have all administrative privileges and the privileges required to execute the business transactions. If someone steals that user ID and password, that person can exercise those privileges from another place in the network. This is a significant exposure.

It also means diminished accountability. For example, there is no ability to separate access performed by the middleware server for its own purposes from those performed on behalf of users.

Characteristics of the common, three-tier architecture include:

- The middle layer (sometimes called the middleware layer) authenticates users running client applications.
- The middle layer also manages interactions with the database server (DB2).
- The middle layer's user ID and password are used for authentication purposes.
- The database privileges associated with that authorization ID are checked when accessing the database, including all access on behalf of all users.

The middle layer's user ID must be granted privileges on all the resources that might be accessed by user requests.

Figure 4-1 shows a common three-tier architecture.

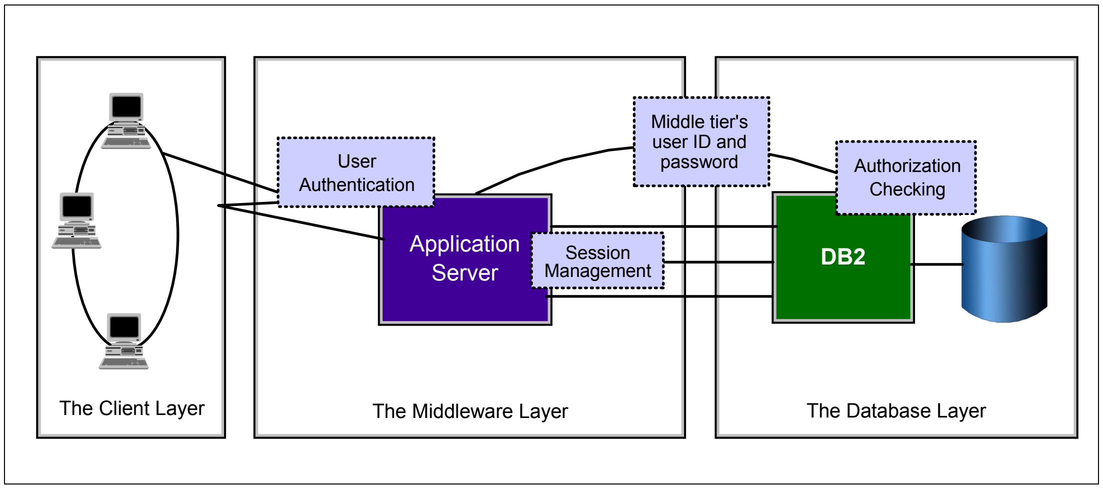


Figure 4-1 Three-tier architecture

4.1.3 Dynamic SQL auditability

Looking at Figure 4-1, we can now discuss the reasons that led to this configuration and the resulting weak auditing capability.

The middleware layer's authid is used to access DB2 so that customers do not have to grant dynamic SQL privileges directly to their users. DB2 only authenticates and knows about a single authid, which make all the requests, when, in fact, there are multiple users initiating requests through the application. A consequence of this approach is diminished accountability; there is no ability in DB2 to separate access performed by the middleware layer's authid server for its own purposes from those performed on behalf of users.

Business user authentication is performed by the middleware component and not in DB2, which means a loss of control over user access to the database.

Another reason for using a three-tiered approach is connection pooling in the application server. There is a performance processing time associated with simply creating a new connection to the database server and re-authenticating the user at the database server. This can be a problem for middleware servers that do not have access to user credentials, such as IBM Lotus® Domino®.

4.1.4 Securing DBA privileges

In this section, we describe some challenges and new opportunities for securing DBA privileges.

Shared SYSADM ID

In some organizations, certain DBAs have powerful user IDs that hold administrative privileges such as SYSADM or DBADM. Although they might only use certain privileges infrequently, they typically hold these privileges all the time, which increases the risk of adversely affecting DB2 objects or table data when an error or mistake occurs. Such a situation is potentially more open to malicious activity or fraud.

Suppose a team of DBAs share a generic user ID that holds SYSADM or DBADM privileges. When a DBA leaves the team for a new internal or external job, there is no cascading effect. However, by using a shared ID, there is no individual accountability, which is a Sarbanes-Oxley requirement.

Additionally, many customers are concerned about DBA access to sensitive data.

Dual responsibilities

A DBA might have a dual job function, where he or she maintains databases in both development and production using the same user ID in both environments. In this scenario, a mistake can occur when making changes if a DBA thinks she is connected to development, when she is really connected to production.

Full-time access to sensitive / private data by a DBA

When DBAs create tables, they have full access to the contents of the tables all of the time.

4.1.5 DBADM create view and drop / alter

An authid, with DBADM privilege, under certain circumstances can create views for others. However, this same authid is not able to drop or grant privileges on these views to others. Because views do not have an underlying database, the DBADM privilege cannot be recognized for views.

4.1.6 Reserving a RACF group and table dropping

When the creator of a table is not a user ID, and a RACF group has not been set up with the same name as the creator, it is possible for someone else to create the group, connect themselves to it, and drop the table. Another user ID can do this with basic access to DB2 and without any grants received on that table.

4.1.7 Exercising granted privileges

Before trusted contexts and roles, privileges held by an authid were universal, irrespective of context. This can weaken security because a privilege can be used for purposes other than those originally intended. For example, if an authid is granted SELECT on a payroll table, the authid can exercise that privilege regardless of how it gains access to the table.

4.2 Roles

DB2 extends the trusted context concept to optionally assign a default role to a trusted context and optionally assign a role to a user of the context.

A role is an independent database entity that groups together one or more privileges and can be assigned to users. A role can provide privileges that are in addition to the current set of privileges granted to the user’s primary and secondary authorization IDs.

A role is only available within a trusted connection.A role can be an object owner.

Roles provide a more flexible technique than groups or users in assigning and controlling authorization, while improving consistency with the industry and improving security.

A database role is a virtual authorization ID that is assigned to an authid through an established trusted connection.

Roles provide a means to acquire context-specific privileges.

A trusted context definition specifies one or more authids that are allowed to use it. As part of *allowing* an authid to use the context, you can also assign a role for each authid.

A role can be used to make privileges or ownership available to different people at different times or to multiple people at the same time.

Roles provide:

- The capability for a DBA to acquire the privileges to create objects even though ownership is by another ID.
- The ability to more finely control when and from where a privilege can be exercised. The role can be assigned and removed from individuals through the trusted context as needed. This situation allows a DBA to perform object maintenance during a change control window on a Saturday night, for example. But when Monday arrives, the role is removed, and the DBA no longer holds the privileges required to do this same work.
- The ability to audit the work completed during the maintenance window, even though the role made this possible. These audit trails are available for verification by a security administrator or auditor. This makes it possible to conclusively establish the identity of those performing the actions rather than the weaker scenario of a shared ID and password with high privileges.
- A mechanism other than authorization IDs through which you can assign privileges and authorities.

4.2.1 Role access to data

A role is recognized and used to control access to data, as shown in Figure 4-2.

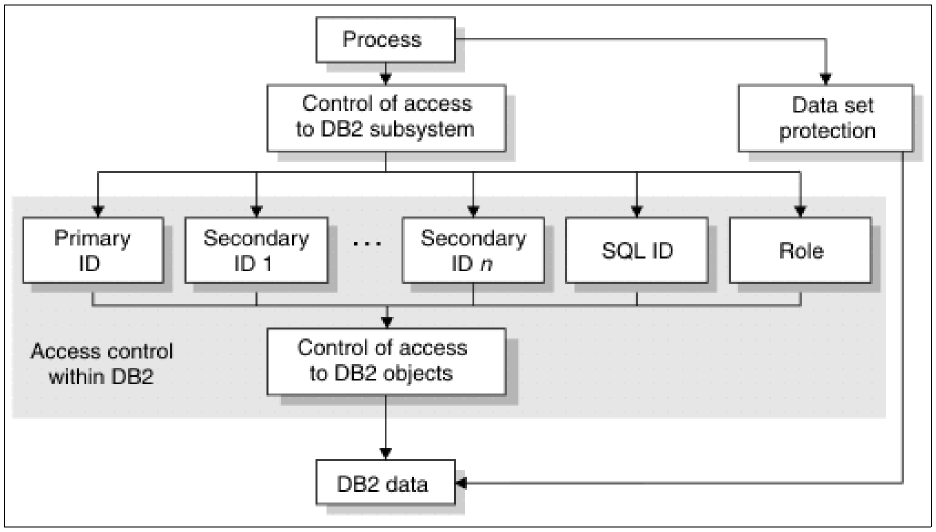

Process
Control of access to DB2 subsystem
Data set protection
Primary ID
Secondary ID 1
...
Secondary ID n
SQL ID
Role
Access control within DB2
Control of access to DB2 objects
DB2 data

Figure 4-2 Access control through a role

4.2.2 Role ownership of objects

Outside trusted contexts and roles, object ownership is tied to a user. When a user creates an object, the user becomes the owner of the object. If that user changes jobs within the company or leaves the company, the object has to be dropped to remove the privileges of that user on the object, and as a result, all grants associated with it are revoked. The object then has to be re-created and the privileges re-granted.

However, if the object owner is a role, removing user privileges does not require the object to be dropped and re-created.

When a role is defined for a trusted context, the role becomes the owner of the objects created in a trusted context if the ROLE AS OBJECT OWNER clause is specified. If a role is defined to be the owner, the role must have all the privileges necessary to create the objects.

The role exists as an object independent of its creator, so creating the role does not produce a dependency on its creator.

If a role owns a created object, the user inheriting the privileges of the role through a trusted context requires a GRANT privilege to access it outside the trusted context.

Role ownership allows more tightly controlled security where DBAs only exercise privileges when they perform approved activities through a trusted context and its role.

If ROLE AS OBJECT OWNER is not specified, object ownership is determined as usual.

4.2.3 Auditing notes

When determining what an authid has access to in DB2, it is important to consider:

- Privileges granted directly to the authid
- Privileges granted to the secondary RACF groups to which this authid belongs (or other security software like ACF2).
- Privileges available to an authid when it is used in a trusted connection
  - Consider any default roles available to this authid in any trusted contexts it is allowed to use.
  - Consider if there is a role assigned directly to this authid in a trusted context definition when it is the system authid.

4.3 Trusted contexts

Trusted contexts can be used to manage access to your DB2 subsystems. Trusted contexts can improve security by controlling how a user connects to DB2.

A trusted context is an independent database entity that you can define based on a system authorization ID and connection trust attributes.

It is likely that only a subset of connection requests for any type and source are trusted or that you want to restrict trusted connections to a specific server.

DB2 V9 introduces a new database entity called a trusted context. It provides a technique to work with other environments more easily than before, improving flexibility and security.

Trusted context addresses the problem of establishing a trusted relationship between DB2 and an external entity, such as a middleware server, for example:

- WebSphere Application Server
- Lotus Domino
- SAP NetWeaver
- Oracle PeopleSoft
- Siebel Optimizer

More granular flexibility allow for the definition of trusted connection objects. When defined, connections from specific users through defined attachments (DDF, RRS Attach, DSN) and source servers allow trusted connections to DB2.

4.3.1 Characteristics of a trusted context

A trusted context is an independent database entity that is based on a system authorization ID and connection trust attributes.

The system authorization ID is a DB2 primary authorization ID that is used to establish the trusted connection. A SYSTEM AUTHID for a connection can only be associated with a single trusted context. This situation ensures that there is always a clear mapping between a specific connection and a specific trusted context.

The connection trust attributes are ADDRESS, SERVAUTH, ENCRYPTION, and JOBNAME. They identify specific connections to consider as part of the trusted context.

Any defined ADDRESS, SERVAUTH, or JOBNAME must also be unique in a trusted context.

A trusted connection can be established for a local or a remote application. The attributes used to establish a trusted context are different for remote versus local applications.

4.3.2 How a trusted connection comes alive

A trusted connection is a database connection that is established when the incoming connection attributes match the attributes of a unique, enabled trusted context defined at the server. The trust attributes identify a set of characteristics about the specific connection that are required for the connection to be considered a trusted connection. The relationship between a connection and a trusted context is established when the connection to the server is first created and that relationship remains for the life of that connection. If a trusted context definition is altered, the changed attributes of the trusted context take effect when the next new connection request comes in, or a switch user request is issued within an existing trusted connection.

4.3.3 Authid switching within a trusted connection

DB2 allows an established trusted connection to be used under a different user. To allow this action, the trusted context must be defined with use for the specific user. If PUBLIC is specified for the user, it allows the trusted connection to be used by any authorization ID. When a trusted connection is established, DB2 enables the trusted connection to be reused under a different user on a transaction boundary. When a switch user event takes place, the new user does not inherit any privileges or resources from the previous user. The trusted connection can be used by a different user with or without authentication. This is specified by the WITH AUTHENTICATION clause. The catalog table SYSIBM.SYSCONTEXTAUTHIDS stores the authorization IDs that can be *switched to* in a trusted connection.

4.4 Challenges addressed by roles and trusted contexts

In this section, we describe how the role and trusted context database entities overcome the challenges highlighted at the beginning of this chapter. For more details, see *Securing DB2 and Implementing MLS on z/OS*, SG24-6480.

4.4.1 Trusting all connection requests

It is likely that only a subset of requests for any type and source are considered trusted. In addition, you might want to restrict trusted connections to one or more specific servers. You can control the source of incoming requests by creating trusted contexts that allow trusted connections to be controlled.

4.4.2 Application server user ID and password (three-tier architecture)

Use trusted context and roles to provide added security for your network-attached application servers and thus limit exposure. These new capabilities allow the DBA to make GRANTs to a role, for example, SAP_ROLE, which can only be used from a specific list of IP addresses. If someone steals the application server's user ID and password, they will not be able to access the database from another place in the network unless they are also able to execute the SQL statement on one of the approved application servers. This protection requires no change to the code in the application server.

4.4.3 Dynamic SQL auditability

The roles and trusted context database entities also enable customers to improve DB2 system auditing without compromising performance. The user's authid can be used to run database transactions, so the DB2 audit is able to identify the users individually (an important capability for meeting some regulatory compliance requirements). User passwords can be optional. The trusted context retains many of the performance benefits of connection pooling with auditability. A key element is the ability to reuse the same physical connection without the need to re-authenticate the user in DB2.

4.4.4 Allowing connections without credentials

With trusted contexts, customers are now able to identify DB2 servers that are trusted to send already verified connection requests. This functionality can be used to establish already verified TCP/IP connections and improves the ability to replace SNA connections with TCP/IP. The communications database is used to identify trusted connections and specify a SYSTEM AUTHID for the trusted context. This makes it possible to automatically propagate the user identity from one DB2 system to another.

Customers can identify DB2 servers that are trusted to send already verified DRDA requests.

4.4.5 Shared SYSADM ID

Trusted contexts and roles can help solve the issues related to sharing the SYSADM ID by enabling an auditable process. You can use the trusted context and role support to implement DBA privileges that can easily be disconnected and reconnected to individual employees.

Privileges can be granted to a role. When the DBA needs to perform a system change, a trusted context is used to assign the role to the DBA temporarily and from a specific batch job or location. A trace is started, the DBA is given the request and executes the database changes, the trace is stopped and the trusted context is disabled, and another DBA reviews the audit trace. This approach provides abilities similar to a shared SYSADM or DBADM user ID, but avoids the known audit compliance problems associated with shared user IDs. The roles can own DB2 objects, so disconnecting a DBA from a role does not cause the objects to be cascade deleted.

With these capabilities, customers can create DBA procedures that can be audited and protected so that one individual cannot violate the established rules without being detected during the audit review.

4.4.6 Dual responsibilities

Although a total segregation of duties eliminates this risk, the new trusted context and role capabilities can be used to reduce this risk by limiting how and when the production environment can be accessed for this DBA, while allowing full privileges and full time access to the development environment.

4.4.7 Full-time access to sensitive/private data

By having roles own objects, even tighter controls can be implemented. Using roles allows customers to move to a model where DBAs have no access to data in production except when they are performing approved scheduled DBA activities. The ability to do those activities can be controlled with a trusted context and a temporary role.

4.4.8 DBADM create view and drop / alter

A trusted context can allow an authid with DBADM or any permitted user to assume the identity of another user, such as the view owner, and then perform the desired actions.

4.4.9 Reserving a RACF group and table dropping

To further protect your tables from this known gap, grant the create table or DBADM privilege solely to a role with role ownership and use the role to create all the tables.

4.4.10 Exercising granted privileges

When granting the UPDATE privilege on a payroll table to an authid, it is a better if this privilege is available to the authid only when it is connected to a computer located inside the company offices. The new trusted context and role capabilities allow the DBA to GRANT privileges, for example, UPDATE, that can only be used from a specified list of IP addresses.

4.5 Example of a local trusted context: Securing DBA activities

Many customers are concerned about DBA access to sensitive customer data. Another common problem is the use of a shared SYSADM user ID or a shared DBADM user ID.

We provide an example of implementing a new application and database that contain private customer information. The implementation adds many tables to a new database called DCUST. Because of the sensitive nature of the data to be stored in this database, the DBAs only have access to the database during the implementation weekend. Afterward, they will have no access at all to the DCUST database.

A user with SYSADM authority executes the DDL in Example 4-1.

Example 4-1 Prepare contexts for implementation weekend

```
CREATE DATABASE DCUST;
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+--
CREATE ROLE PROD_CUST_DBA_ROLE ;
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+--
GRANT DBADM ON DATABASE DCUST TO ROLE PROD_CUST_DBA_ROLE ;
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+--
 CREATE TRUSTED CONTEXT CTXT_PROD_CUST_DBA_ROLE_USRT029
 BASED UPON CONNECTION USING SYSTEM AUTHID USRT029
 DEFAULT ROLE PROD_CUST_DBA_ROLE WITH ROLE AS OBJECT OWNER AND QUALIFIER
 ATTRIBUTES (JOBNAME 'USRT029')
 DISABLE;
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+--
 CREATE TRUSTED CONTEXT CTXT_PROD_CUST_DBA_ROLE_USRT039
 BASED UPON CONNECTION USING SYSTEM AUTHID USRT039
 DEFAULT ROLE PROD_CUST_DBA_ROLE WITH ROLE AS OBJECT OWNER AND QUALIFIER
 ATTRIBUTES (JOBNAME 'USRT039')
 DISABLE;
```

On the Saturday morning of the implementation weekend, a user with SYSADM authority enables the trusted contexts for use by the production DBAs, as shown in Example 4-2.

Example 4-2 Enabling the trusted contexts for use by the production DBAs

```
ALTER TRUSTED CONTEXT CTXT_PROD_CUST_DBA_ROLE_USRT029
ALTER ENABLE ;
---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+-------
ALTER TRUSTED CONTEXT CTXT_PROD_CUST_DBA_ROLE_USRT039
ALTER ENABLE ;
---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+-------
```

```
---------+---------+---------+---------+---------+-------
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
```

At the end of the weekend, both trusted contexts are disabled and the DBAs no longer have the DBADM authority on database DCUST.

For more details, see Chapter 8, "Network trusted contexts and roles", in *Securing DB2 and Implementing MLS on z/OS*, SG24-6480.

4.6 Example of a remote trusted connection

The HR team hires a new person. This person will use authid PAOLOR7 and will have access to the tightly controlled employee table containing all the salaries. PAOLOR7 is on probation for the first 6 months and must use the office desktop computer to access the table.

In Example 4-3, DB2R2 creates the context that allows PAOLOR7 to access the employee table using TSO only.

Example 4-3 Create a trusted context for new hire

```
CREATE TRUSTED CONTEXT TC_PAOLOR7
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR7
DEFAULT ROLE HRIBM
ATTRIBUTES (JOBNAME 'PAOLOR7')
ENABLE;
---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

In Example 4-4, we show that PAOLOR7 has access to the data.

Example 4-4 The employee table data

```
SELECT * FROM R9.EMPL;
---------+---------+---------+---------+---------+---------+
EMPLID  DEPTID                  SALARY
---------+---------+---------+---------+---------+---------+
111111  ACT                      50000
222222  ACT                      60000
333333  SAL                      70000
444444  DEV                      80000
555555  SUP                      90000
666666  PUR                     100000
777777  PUR                     110000
888881  ACT                     120000
888882  SAL                     130000
888883  DEV                     140000
999999  FIN                     150000
DSNE610I NUMBER OF ROWS DISPLAYED IS 11
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

The employee table updates are done on the first day of the month. The next update is scheduled for Sunday. On the Friday prior to the scheduled update, the person normally doing the updates (PAOLOR8) wins the lottery and no longer works for the company.

PAOLOR7 is at home enjoying the weekend when the manager calls him to ask if he could do the updates now as it late in the afternoon on Sunday. PAOLOR7 is still on probation and he only has access through his office desktop. The manager of PAOLOR7 has another computer with a DB2 client installed.

PAOLOR7 has a lap top with the DB2 client installed but no PCOM software. The manager asks PAOLOR7 for his IP address and then asks DB2R2 to alter the trusted context by completing the following steps:

1. DB2R2 (SYSADM) recreates the trusted context shown in Example 4-5.

   Example 4-5 Create a remote trusted context for PAOLOR7

   ```
   CREATE TRUSTED CONTEXT TC_PAOLOR7
     BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR7
     DEFAULT ROLE HRIBM
     ATTRIBUTES (ADDRESS '9.30.28.115')
     ENABLE;
   ---------+---------+---------+---------+---------+---------+-
   DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
   ```

2. A grant of the update privilege is also required, which requires the following SQL run by DB2R2:

   ```
   GRANT UPDATE ON R9.EMPL TO ROLE HRIBM;
   ```

3. PAOLOR7 now connects from home using the DB2 client and a trusted connection. In Example 4-6, we display the results of a display thread command.

   Example 4-6 Display the trusted connection

   ```
   485-TRUSTED CONTEXT=TC_PAOLOR7,
       SYSTEM AUTHID=PAOLOR7,
       ROLE=HRIBM
   ```

4. PAOLOR7 acquires the HRIBM role and is able to remotely access and update the employee table.

4.7 Example of a remote trusted connection with multiple users

This scenario is covered in detail in Chapter 7, “User authentication” on page 125. More information is also included in Chapter 9, “A WebSphere implementation”, in *Securing DB2 and Implementing MLS on z/OS*, SG2-6480. Here we show only a high level description of this solution.

This is a distributed application using IBM WebSphere Application Server Version 6.1 with DB2 9 for z/OS. We use the following elements for a solution:

- A trusted context using roles
- The propagation of user IDs from the client to DB2

  We use Enterprise Identity Mapping (EIM) map user IDs with authorization IDs on DB2. For information about EIM, see *DB2 10 for z/OS Administration Guide,* SC19-2968.
- Secure Sockets Layer (SSL) encryption instead of DRDA encryption

The goals are to:

- Limit the access to the DB2 server by limiting the authorization IDs belonging to the application to a single IP address.
- Improve auditing of users by being able to pinpoint which user did what.
- Ensure the highest available encryption on distributed connections to DB2.

The application runs in a typical three-tier environment. We have DB2 9 for z/OS acting as a database server running on z/OS V1.7. The application is a Java servlet running in WebSphere Application Server V6.1 on a Microsoft Windows server, connecting to DB2 using a type 4 Universal Java driver. The application is invoked through any Internet browser.

Figure 4-3 shows the architecture of this application.

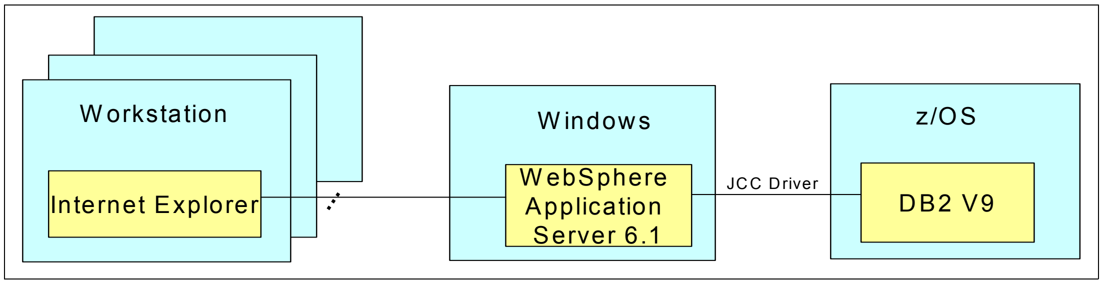


Figure 4-3 Three-tiered application model

Example 4-7 shows the trusted context used for this scenario.

Example 4-7 Trusted context for remote WebSphere Application Server implementation

```
CREATE TRUSTED CONTEXT WASCTXT1
BASED UPON CONNECTION USING SYSTEM AUTHID ADMF001
ATTRIBUTES (ADDRESS '9.30.30.207')
WITH USE FOR USRT001, USRT002, USERT003
ENABLE;
```

The trusted context is named WASCTXT1 and it is reachable from 9.30.30.207, the IP address of the Windows server. The system authorization ID that is allowed to establish a connection through the trusted context is ADMF001, matching the user name entered previously in the Java authentication parameters of WebSphere Application Server.

When the trusted connection is established, the users USRT001, USRT002, and USRT003 are allowed to use the connection by switching the user ID. We use the default for AUTHENTICATION, which is no authentication, meaning that no password is necessary when switching to a new user.

Finally, we create RACF users with the same names as the Windows user IDs. Even though the user ID and password are not authenticated, RACF still needs to confirm the existence of the user ID and assign primary and secondary authorization IDs and CURRENT SQLID.

For more information about this solution, see *Securing DB2 and Implementing MLS on z/OS*, SG24-6480.

4.8 Protecting new DB2 10 administrative authorities

in this section, we provide examples of using some of the new DB2 10 administrative authorities.

4.8.1 SYSTEM DBADM

In this section, we show an example for development and an example for production.

In this environment, the development databases do not contain any production data, even for a limited time. In this environment, there are three DBAs in the development team. We create a shared DBA development role and enable the people on the team to connect only through TSO to DB2. We create the role and trusted contexts shown in Example 4-8.

Example 4-8 SYSTEM DBADM in development

```
CREATE ROLE ROLE_DEV_DBA;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
GRANT DBADM WITH DATAACCESS WITH ACCESSCTRL
      ON SYSTEM TO ROLE ROLE_DEV_DBA;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
------------------------------------------------------
CREATE TRUSTED CONTEXT TC_DEV_DBA_PAOLOR7
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR7
DEFAULT ROLE ROLE_DEV_DBA WITH ROLE AS OBJECT  OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLOR7')
ENABLE;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
CREATE TRUSTED CONTEXT TC_DEV_DBA_PAOLOR8
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR8
DEFAULT ROLE ROLE_DEV_DBA WITH ROLE AS OBJECT  OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLOR8')
ENABLE;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
CREATE TRUSTED CONTEXT TC_DEV_DBA_PAOLOR9
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR9
DEFAULT ROLE ROLE_DEV_DBA WITH ROLE AS OBJECT  OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLOR9')
ENABLE;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

For the production environment, we create a shared production role with no ability to grant privileges on any objects, including the ones created by this role. Example 4-9 shows how this production environment is set up.

Example 4-9 Setup for DBAs in production

```
CREATE ROLE ROLE_PROD_DBA;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
GRANT DBADM WITH DATAACCESS WITHOUT ACCESSCTRL
      ON SYSTEM TO ROLE ROLE_PROD_DBA;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
------------------------------------------------------
CREATE TRUSTED CONTEXT TS_PROD_DBA_PAOLORA
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLORA
DEFAULT ROLE ROLE_PROD_DBA WITH ROLE AS OBJECT  OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLORA')
ENABLE;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
CREATE TRUSTED CONTEXT TS_PROD_DBA_PAOLORB
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLORB
DEFAULT ROLE ROLE_PROD_DBA WITH ROLE AS OBJECT  OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLORB')
ENABLE;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

We grant access control to a role that is only available to two security administrators and only through TSO, as shown in Example 4-10.

Example 4-10 Granting access control to a role

```
------------------------------------------------------                0006000
CREATE ROLE ROLE_PROD_ACCESS;                                        00007000
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
GRANT ACCESSCTRL ON SYSTEM TO ROLE ROLE_PROD_ACCESS;                 00008015
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE TRUSTED CONTEXT TC_PROD_PERF_PAOLOR3                          00009001
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR3                    00010001
DEFAULT ROLE ROLE_PROD_ACCESS                                        00020000
ATTRIBUTES (JOBNAME 'PAOLOR3')                                       00030001
ENABLE;                                                              00040000
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE TRUSTED CONTEXT TC_PROD_PERF_PAOLOR5                          00050000
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR5                    00060000
```

```
DEFAULT ROLE ROLE_PROD_ACCESS                                           00070000
ATTRIBUTES (JOBNAME 'PAOLOR5')                                          00080000
ENABLE;                                                                 00090000
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
COMMIT;                                                                 00090100
---------+---------+---------+---------+---------+---------+---------+---------+
```

Attention: If a role or authid has DBADM WITHOUT DATAACCESS on SYSTEM, that role or authid will still have access to data in all tables and objects that it creates.

4.8.2 SECADM

The SECADM authority is not grantable. With SEPARATE_SECURITY set to YES, the SECADM authority is assigned using the four values specified in the following DSNZPARMs:

- SECADM1 =
- SECADM1_TYPE=
- SECADM2=
- SECADM2_TYPE=

With SEPARATE_SECURITY set to NO, users with the SYSADM authority implicitly have the SECADM authority.

Example 4-11 shows an example where the assignment of the SECADM1 is made to a role.

Example 4-11 SECADM assigned to a role

```
SECADM1=ROLE_PROD_SECADMADM
SECADM1_TYPE=ROLE,
```

Note that the SECADM authority is associated to the role only through this DSNZPARM; there is no grant, as the SECADM privilege is not grantable. It is a best practice to create the roles and trusted contexts before switching to SEPARATE_SECURITY set to YES. Consider using one SECADM authority assigned to a RACF group authid and the other to a role.

There are initially two people doing the security administration in production. Example 4-12 shows a sample setup of the trusted contexts creation for these two security administrators.

Example 4-12 Trusted context creation for two security administrators

```
CREATE ROLE ROLE_PROD_SECADM;
---------+---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+-
CREATE TRUSTED CONTEXT SEC_TEAM_PAOLOR8
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR8
DEFAULT ROLE ROLE_PROD_SECADM WITH ROLE AS OBJECT OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLOR8')
ENABLE;
```

4.8.3 SQLADM

The SQLADM authority allows you to issue the SQL EXPLAIN statements, execute the PROFILE commands, run the RUNSTATS and MODIFY STATISTICS utilities on all user databases, and execute system-defined routines, such as stored procedures or functions, and any packages that are executed within the routines. It also has MONITOR1 and MONITOR2 system privileges.

In DB2 10, it is now possible to explain SQL statements without having the privilege to access the data. Consider granting this privilege to production people working with SQL and performance. Be aware that MONITOR2 is powerful, as it can receive all trace data.

In Example 4-13, we show a sample setup with two persons in a production performance team who are allowed to exercise the SQLADM on system authority from TSO only.

Example 4-13 Make SQLADM available

```
CREATE ROLE ROLE_PROD_SQL;
---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+--------
GRANT SQLADM ON SYSTEM TO ROLE_PROD_SQL;
---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+--------
CREATE TRUSTED CONTEXT TC_PROD_PERF_PAOLOR6
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR6
DEFAULT ROLE ROLE_PROD_SQL
ATTRIBUTES (JOBNAME 'PAOLOR6')
ENABLE;
---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+--------
CREATE TRUSTED CONTEXT TC_PROD_PERF_PAOLOR7
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR7
DEFAULT ROLE ROLE_PROD_SQL
ATTRIBUTES (JOBNAME 'PAOLOR7')
ENABLE;
---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

5

Data access control

DB2 10 provides new options to data control access that allows for more granularity and additional flexibility. You can manage SQL access to a table at the level of column, row, or both.

These functions make it possible to build a system model controlling untrusted SQL applications. Accesses to data for administration and utilities execution need to be controlled with traditional functions.

This chapter contains the following topics:

- New access control functions and terminology
- Row permission object
- Column masks
- EXPLAIN table information
- Triggers and UDF information

5.1 New access control functions and terminology

Prior to DB2 10, when you wanted to control access at the row and column level, you normally used views. This situation usually works for simple views, but becomes ineffective when view definitions become too complex. It also becomes costly when a large number of views must be manually updated and maintained.

This dilemma can now be satisfied by using an SQL solution managed by DB2 security administrators. There is no need to manage many views or use view updates.

Row and column access control is based on a security policy that specifies the rules and conditions under which a user, group, or role can access rows, columns, or both of a base table. The control policies can be added, modified, or removed to meet current company rules without changing applications.

If the system parameter SEPARATE_SECURITY is set to YES, you need a user ID or role with SECADM authority to create row permissions and columns masks. This authority could activate or deactivate row and column access control for a table.

If the system parameter SEPARATE_SECURITY is set to NO, the SYSADM authority can perform the same task.

Before we discuss the new DB2 10 access control features in detail, we clarify the new terminology for these new controls and objects:

Access control	The method that DB2 10 introduces to implement access control at the row and at column level.
Row and column access control	This method is based on a security policy that specifies the rules and conditions under which a user, group, or role can access the rows, the columns, or both of a table.
Row access control	The DB2 security mechanism that uses SQL to control access to a table at row level. You can activate row access control through ALTER TABLE DDL.
Column access control	The DB2 security mechanism that uses SQL to control access to a table at column level. You activate column access control through ALTER TABLE DDL.
Row permission	A DB2 object that describes the row filtering rule DB2 implicitly applies to all table rows for every user whenever the table is accessed using SQL DML You create a row permission by the new CREATE PERMISSION DDL.
Column mask	A DB2 object that describes the column masking rule DB2 implicitly applies to a table column for every user whenever the table column is referenced using SQL DML. You create a column mask by the new CREATE MASK DDL.

5.2 Row permission object

A row permission is a DB2 object that describes a specific row access control rule for a table.

If the SEPARATE_SECURITY system parameter on panel DSNTIPP1 is set to YES during installation or migration, you must have the SECADM authority to create a row permission. If SEPARATE_SECURITY is set to NO, you must have the SECADM or SYSADM authority,

You can create as many row permissions as you want and enable these objects any time. DB2 enforces these objects types only if you explicitly enable them through ALTER PERMISSION DDL and if you alter the table through ALTER TABLE DDL to activate row access control.

The new create command to define permissions is shown in Figure 5-1.

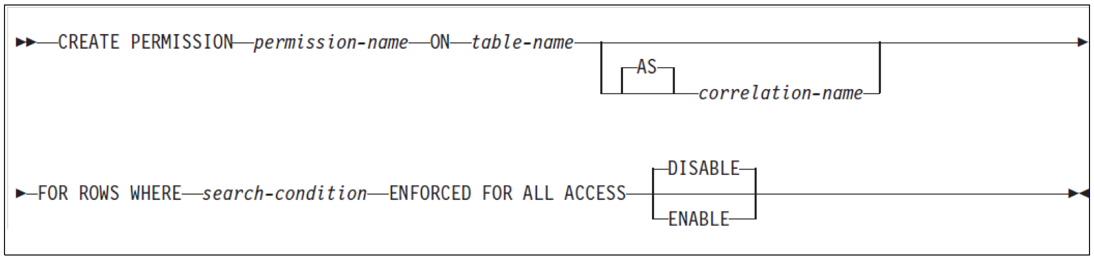


Figure 5-1 Create permission DDL

The two parameters offer the following options:

DISABLE Specifies that the row permission will be disabled for row access control. The row permission remains ineffective whether the row access control is activated from the table or not.

ENABLE Specifies that the row permission will be enabled for row access control. If row access control is not currently activated for the table, the row permission becomes effective when row access control is activated for the table. If row access control is currently activated for the table, the row permission becomes effective immediately and all packages and dynamic cached statements that reference the table are invalidated.

You can see details about the *search condition* in *DB2 10 for z/OS SQL Reference,* SC19-2983 and example of this conditions in Chapter 10, "Implementing data access control" on page 211.

Figure 5-2 shows the alter command for the row permission objects.

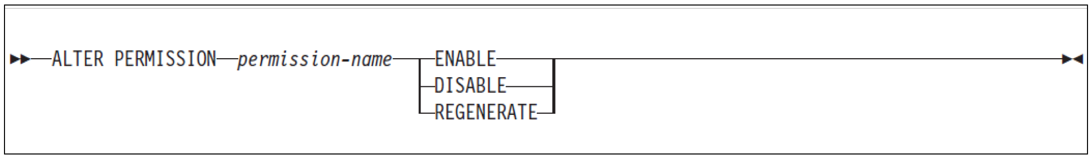


Figure 5-2 Alter permission DDL

The REGENERATE parameter specifies that the row permission will be regenerated. The row permission definition in the catalog is used, and existing authorizations and dependencies if any, are retained. When DB2 maintenance is applied that affects how a row permission is generated, the row permission might need to regenerated to ensure the row permission is still valid.

To support the activation and deactivation of row access control, the ALTER TABLE DDL statement is extended, as shown in Figure 5-3.

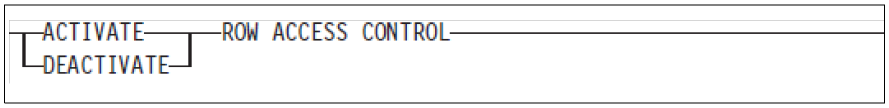


Figure 5-3 Alter table row access control

If row access control is currently activated for the table, the row permissions becomes effective immediately and all packages and dynamic cached statements that reference the table are invalidated. Prior to activating the change, you want to know the packages and dynamic statement cache entries that the change invalidates. To determine the packages that is invalidated when the row permission becomes active for a table, you can query the SYSIBM.SYSPACKDEP table, as shown in Example 5-1.

Example 5-1 Query SYSIBM.SYSPACKDEP

```
SELECT (BQUALIFIER !! '.'!! BNAME) AS TABLE_NAME ,
(DCOLLID !! '.'!! DNAME) AS COLLID_PACKAGEID
FROM SYSIBM.SYSPACKDEP WHERE
BQUALIFIER = 'GLWSAMP' AND BNAME = 'LCNSEL'
-------------------------------------------

TABLE_NAME     COLLID_PACKAGEID
 *              *
 -------------- ----------------
 GLWSAMP.LCNSEL GLWSAMP.DPTUPR
 GLWSAMP.LCNSEL GLWSAMP.DPTADD
 GLWSAMP.LCNSEL GLWSAMP.DPTUPD
```

You can create multiple row permissions. The row permission remains ineffective until you activate it. The policies implemented by the row permission apply to accesses for SELECT,INSERT, UPDATE,DELETE, and MERGE. They do not apply to the UNLOAD, REORG pause, and DSN1COPY utilities.

Attention: During table access, DB2 transparently applies these rules to every user, including the table owner and the install SYSADM, SECADM, SYSTEM DBADM, and DBADM authorities.

5.2.1 Built-in functions

DB2 provides the following scalar built-in functions to support row access control:

- VERIFY_GROUP_FOR_USER
- VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER
- VERIFY_ROLE_FOR_USER

You can use these built-in functions to verify from a row permission whether the SQL user is connected to a RACF group, whether the user runs within a given trusted context, and whether the user runs under a given role. These built-in functions can help you control row permissions based on RACF groups, trusted contexts, and roles.

Although these built-in functions are introduced to support row access control, you can use them outside of row permission and column mask objects as well. For more information about these new built-in functions, refer to *DB2 10 for z/OS SQL Reference,* SC19-2983.

5.2.2 Creating and activating a row permission

If the system parameter SEPARATE_SECURITY is set to YES, SECADM authority can activate row permission access control against a table by executing one alter table statement. If SEPARATE_SECURITY is set to NO, you must have the SECADM or SYSADM authority,

Row permission access control can be activated before any row permission policy is created.

Note: Notice that as soon as row permission access control is activated, DB2 implicitly creates a system default row permission policy, which inhibits any access to that table data (row). The access to that table data (row) is permitted only by creating and enabling at least one row permission policy. The number of policies to be created and enabled depend on your business rules and the security goal that you need to accomplish.

Scenario for creating a row permission

In this section, we provide an example of a row permission. The scenario has the following components:

- Table SPF.EMP is created by DBA user ID PAOLOR5, who has full access to the table's data.
- User ID PAOLOR2 has *SECADM authority* under the SECHEAD RACF group.
- User ID PAOLOR6 is a manager for the D21 and E21 departments and belongs to the MANAGER RACF group.
- The system parameter is SEPARATE_SECURITY=YES.

This scenario requires the completion of the following steps:

1. User ID PAOLOR5(DBA) issues a SELECT command over the table and receives the result shown in Example 5-2.

Example 5-2 Output for an authorized user

```
SELECT * FROM SPF.EMP

-----------------------------------

******************************** Top of Data ****************************
 EMPNO   FIRSTNME  MIDINIT LASTNAME   WORKDEPT PHONENO HIREDATE   JOB
 ------- --------- ------- ---------- -------- ------- ---------- --------
 PAOLOR8 CHISTINE  I       HAAS       A00      3978    1965-01-01 DIRECTOR
 000020  MICHAEL   L       THOMPSON   B01      3476    1973-10-10 MANAGER
 000030  SALLY     A       KWAN       C01      4738    1975-04-05 MANAGER
 000050  JOHN      B       GEYER      E01      6789    1949-08-17 MANAGER
 000060  IRVING    F       STERN      D11      6423    1973-09-14 MANAGER
 PAOLOR6 EVA       D       PULASKI    D21      7831    1980-09-30 MANAGER
 PAOLOR3 EILEEN    W       HENDERSON  E11      5498    1970-08-15 PROJL
 000100  THEODORE  Q       SPENSER    E21      0972    1980-06-19 MANAGER
```

2. User ID PAOLOR2(SECADM) issues a SELECT command over the table and receives the result shown in Example 5-3.

Example 5-3 Output for a non-authorized user

```
   SELECT * FROM SPF.EMP;

-------------------------------------

Rollback done
   SQLCODE : -551                           DSNTIAR CODE :  0

DSNT408I SQLCODE = -551, ERROR:  SECHEAD DOES NOT HAVE THE PRIVILEGE TO PERFORM
         OPERATION SELECT ON OBJECT SPF.EMP
```

3. User ID PAOLOR2(SECADM) issues an ALTER TABLE command and receives the results shown in SYSIBM.SYSTABLES, as shown in Example 5-4, and the results shown in SYSIBM.SYSCONTROLS, as shown in Example 5-5.

Example 5-4 Effect of ALTER TABLE SPF.EMP on SYSIBM.SYSTABLES

```
ALTER TABLE SPF.EMP ACTIVATE ROW ACCESS CONTROL

-------------------------------------

SELECT SUBSTR(NAME,1,10)AS NAME,CONTROL FROM SYSIBM.SYSTABLES
   WHERE NAME = 'EMP'
-------------------------------------

---------+---------+---------+---------+---------+---------+-----
NAME        CONTROL
---------+---------+---------+---------+---------+---------+-----
EMP         R
```

Example 5-5 Effect of ALTER TABLE SPF.EMP on SYSIBM.SYSCONTROLS

```
SELECT SCHEMA,NAME,OWNER,RULETEXT
FROM SYSIBM.SYSCONTROLS
WHERE TBNAME = 'EMP' AND CONTROL_TYPE ='R'
-------------------------------------

------ ------------------------------ ------ --------
SCHEMA NAME                           OWNER  RULETEXT
------ ------------------------------ ------ --------
SPF    SYS_DEFAULT_ROW_PERMISSION__EMP SYSIBM 1=0
```

4. The DDL in for SYS_DEFAULT_ROW_PERMISSION__EMP is executed, as shown in Example 5-6.

Example 5-6 Creating and enabling a row permission

```
CREATE PERMISSION SPF.SYS_DEFAULT_ROW_PERMISSION__EMP ON SPF.EMP
     FOR ROWS
     WHERE 1 = 0
     ENFORCED FOR ALL ACCESS
     ENABLE ;
```

User ID PAOLOR5(DBA) issues a SELECT command over table SPF.EMP and receives the result shown in Example 5-7.

Example 5-7 Row permission exists and is enabled

```
SELECT * FROM SPF.EMP;

---------------------------------
---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME     MIDINIT  LASTNAME        WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

5. User ID PAOLOR2(SECADM) issues a CREATE ROW PERMISSION DISABLE command, as shown in Example 5-8.

Example 5-8 Disabling the row permission

```
CREATE PERMISSION SECHEAD.ROW_EMP_RULES ON SPF.EMP
  FOR ROWS
  WHERE (VERIFY_GROUP_FOR_USER(SESSION_USER, 'MANAGER') = 1
    AND WORKDEPT IN('D21', 'E21'))
  ENFORCED FOR ALL ACCESS
  DISABLE ;
```

6. We verify the output from SYSIBM.SYSCONTROLS using the query and results shown in Example 5-9.

Example 5-9 Results of disablement in SYSIBM,SYSCONTROL

```
SELECT SCHEMA,NAME,OWNER,RULETEXT
FROM SYSIBM.SYSCONTROLS
WHERE TBNAME = 'EMP' AND CONTROL_TYPE ='R'

-------------------------------------
SCHEMA  NAME                           OWNER   RULETEXT
*       *                              *       *
------- ------------------------------ ------- -----------------------------
SPF     SYS_DEFAULT_ROW_PERMISSION__EMP SYSIBM  1=0
SECHEAD ROW_EMP_RULES                  SECHEAD (VERIFY_GROUP_FOR_USER(SESSIO
```

7. User ID PAOLOR6(MANAGER) issues a SELECT command over the table, as shown in Example 5-10.

Example 5-10 Row permission exists as disabled

```
   SELECT * FROM SPF.EMP;

---------------------------------


---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME     MIDINIT  LASTNAME        WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
```

8. User ID PAOLOR2(SECADM) issues an ALTER ROW PERMISSION command to enable it:

```
ALTER PERMISSION ROW_EMP_RULES ENABLE;
```

9. User ID PAOLOR6(MANAGER) issues a SELECT command over the table SPF.EMP and receives the results shown in Example 5-11.

Example 5-11 Row permission enabled

```
SELECT * FROM SPF.EMP;

-------------------------------------

---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME      MIDINIT  LASTNAME         WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
PAOLOR6  EVA           D        PULASKI          D21       7831     1980-09-30
000100   THEODORE      Q        SPENSER          E21       0972     1980-06-19
000230   JAMES         J        JEFFERSON        D21       4200     1966-11-21
000240   SALVATORE M       MARINO           D21       3780     1979-12-05
000250   DANIEL        S        SMITH            D21       0961     1969-10-30
000260   SYBIL         V        JOHNSON          D21       8953     1975-09-11
000270   MARIA         L        PEREZ            D21       9001     1980-09-30
000320   RAMLAL        V        MEHTA            E21       9990     1965-07-07
DSNE610I NUMBER OF ROWS DISPLAYED IS 8
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+--------
---------+---------+---------+---------+---------+---------+---------+--------
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
```

10. User ID PAOLOR2(SECADM) issue a SELECT command over the table and receives the results shown in Example 5-12.

Example 5-12 Results of no rows for SECADM select

```
SELECT * FROM SPF.EMP;
Rollback done
   SQLCODE : -551                          DSNTIAR CODE :  0

DSNT408I SQLCODE = -551, ERROR:  SECHEAD DOES NOT HAVE THE PRIVILEGE TO PERFORM
         OPERATION SELECT ON OBJECT SPF.EMP
```

Scenario with SEPARATE_SECURITY=NO

We now illustrate a scenario that has the following components:

- Table SPF.EMP is created by the DBA user ID PAOLOR5, who has full access to t he table's data.
- User ID DB2R3 has *SYSADM authority*.
- User ID PAOLOR6 is the manager for D21 and E21 departments and belongs to the MANAGER RACF group.
- The system parameter SEPARATE_SECURITY=NO.

This scenario requires the completion of the following steps:

1. User ID PAOLOR5(DBA) issues a SELECT command over the table and receives the results shown in Example 5-13.

   Example 5-13 Results for authorized user

```
SELECT * FROM SPF.EMP

------------------------------------

********************************* Top of Data *****************************
 EMPNO   FIRSTNME  MIDINIT LASTNAME   WORKDEPT PHONENO HIREDATE   JOB
 ------- --------- ------- ---------- -------- ------- ---------- --------
 PAOLOR8 CHISTINE  I       HAAS       A00      3978    1965-01-01 DIRECTOR
 000020  MICHAEL   L       THOMPSON   B01      3476    1973-10-10 MANAGER
 000030  SALLY     A       KWAN       C01      4738    1975-04-05 MANAGER
 000050  JOHN      B       GEYER      E01      6789    1949-08-17 MANAGER
 000060  IRVING    F       STERN      D11      6423    1973-09-14 MANAGER
 PAOLOR6 EVA       D       PULASKI    D21      7831    1980-09-30 MANAGER
 PAOLOR3 EILEEN    W       HENDERSON  E11      5498    1970-08-15 PROJL
 000100  THEODORE  Q       SPENSER    E21      0972    1980-06-19 MANAGER
```

2. User ID DB2R3(SYSADM) issues a SELECT command over the table and receives the result shown in Example 5-14.

   Example 5-14 Results for SYSADM

```
SELECT * FROM SPF.EMP;

------------------------------------

********************************* Top of Data *****************************
 EMPNO   FIRSTNME  MIDINIT LASTNAME   WORKDEPT PHONENO HIREDATE   JOB
 ------- --------- ------- ---------- -------- ------- ---------- --------
 PAOLOR8 CHISTINE  I       HAAS       A00      3978    1965-01-01 DIRECTOR
 000020  MICHAEL   L       THOMPSON   B01      3476    1973-10-10 MANAGER
 000030  SALLY     A       KWAN       C01      4738    1975-04-05 MANAGER
 000050  JOHN      B       GEYER      E01      6789    1949-08-17 MANAGER
 000060  IRVING    F       STERN      D11      6423    1973-09-14 MANAGER
 PAOLOR6 EVA       D       PULASKI    D21      7831    1980-09-30 MANAGER
 PAOLOR3 EILEEN    W       HENDERSON  E11      5498    1970-08-15 PROJL
 000100  THEODORE  Q       SPENSER    E21      0972    1980-06-19 MANAGER
```

3. User ID DB2R3(SYSADM) issues the ALTER TABLE command:

```
ALTER TABLE SPF.EMP ACTIVATE ROW ACCESS CONTROL
```

   The output from SYSIBM.SYSTABLES is shown in Example 5-15.

   Example 5-15 Contents of SYSIBM.SYSTABLES after row control activation

```
SELECT SUBSTR(NAME,1,10)AS NAME,CONTROL FROM SYSIBM.SYSTABLES
   WHERE NAME = 'EMP'

-------------------------------------

---------+---------+---------+---------+---------+---------+-----
NAME        CONTROL
```

```
---------+---------+---------+---------+---------+---------+-----
EMP        R
```

The output from SYSIBM.SYSCONTROLS is shown in Example 5-16.

Example 5-16 Contents of SYSIBM.SYSTCONTROLS after row control activation

```
SELECT SCHEMA,NAME,OWNER,RULETEXT
FROM SYSIBM.SYSCONTROLS
WHERE TBNAME = 'EMP' AND CONTROL_TYPE ='R'

------------------------------------

.------ ------------------------------ ------ --------
SCHEMA NAME                            OWNER  RULETEXT
------ ------------------------------ ------ --------
SPF    SYS_DEFAULT_ROW_PERMISSION__EMP SYSIBM 1=0
```

4. The DDL for SYS_DEFAULT_ROW_PERMISSION__EMP is shown in Example 5-17.

Example 5-17 Creating and enabling a row permission

```
CREATE PERMISSION SPF.SYS_DEFAULT_ROW_PERMISSION__EMP ON SPF.EMP
  FOR ROWS
  WHERE 1 = 0
  ENFORCED FOR ALL ACCESS
  ENABLE ;
```

5. User ID DB2R3(SYSADM) or PAOLOR5(DBA) issues a SELECT command over the table and receives the result shown in Example 5-18.

Example 5-18 Results from SELECT for DBA and SYSADMr

```
SELECT * FROM SPF.EMP;

------------------------------------

---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME     MIDINIT  LASTNAME        WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

6. User ID DB2R3(SYSADM) issues a CREATE ROW PERMISSION DISABLE command, as shown in Example 5-19.

Example 5-19 SYSADM disabling a row permission

```
CREATE PERMISSION SECHEAD.ROW_EMP_RULES ON SPF.EMP
  FOR ROWS
  WHERE (VERIFY_GROUP_FOR_USER(SESSION_USER, 'MANAGER') = 1
    AND WORKDEPT IN('D21', 'E21'))
  ENFORCED FOR ALL ACCESS
  DISABLE ;
```

The output from SYSIBM.SYSCONTROLS is shown in Example 5-20.

Example 5-20 Results from checking SYSIBM.SYSCONTROLS

```
SELECT SCHEMA,NAME,OWNER,RULETEXT
FROM SYSIBM.SYSCONTROLS
WHERE TBNAME = 'EMP' AND CONTROL_TYPE ='R'

------------------------------------

SCHEMA  NAME                           OWNER   RULETEXT
*       *                              *       *
------- ------------------------------ ------- ----------------------------
SPF     SYS_DEFAULT_ROW_PERMISSION__EMP SYSIBM  1=0
SECHEAD ROW_EMP_RULES                  SECHEAD (VERIFY_GROUP_FOR_USER(SESSIO
```

7. User ID PAOLOR6(MANAGER) or DB2R3(SYSADM) issues a SELECT command over the table and receives the result shown in Example 5-21.

Example 5-21 Results from select with row permission disabled

```
SELECT * FROM SPF.EMP;

---------------------------------

---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME     MIDINIT  LASTNAME        WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
```

8. User ID DB2R3(SYSADM) issues the following ALTER PERMISSION to enable the row permission:

   `ALTER PERMISSION ROW_EMP_RULES ENABLE;`

9. User ID PAOLOR6(MANAGER) issues a SELECT command over the table and receives the result shown in Example 5-22.

Example 5-22 Results for manager from select after row permission enabled

```
SELECT * FROM SPF.EMP;

---------------------------------

---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME     MIDINIT  LASTNAME        WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
PAOLOR6  EVA          D        PULASKI         D21       7831     1980-09-30
000100   THEODORE     Q        SPENSER         E21       0972     1980-06-19
000230   JAMES        J        JEFFERSON       D21       4200     1966-11-21
000240   SALVATORE    M        MARINO          D21       3780     1979-12-05
000250   DANIEL       S        SMITH           D21       0961     1969-10-30
000260   SYBIL        V        JOHNSON         D21       8953     1975-09-11
000270   MARIA        L        PEREZ           D21       9001     1980-09-30
000320   RAMLAL       V        MEHTA           E21       9990     1965-07-07
DSNE610I NUMBER OF ROWS DISPLAYED IS 8
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

```
---------+---------+---------+---------+---------+---------+---------+--------
---------+---------+---------+---------+---------+---------+---------+--------
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
```

10. User ID PAOLOR5(DBA) or DB2R3(SYSADM) issues a SELECT command over table SPF.EMP and receives the result shown in Example 5-23.

Example 5-23 Results (no rows) from select for DBA and SYSADM

```
SELECT * FROM SPF.EMP;

-------------------------------------

---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME      MIDINIT  LASTNAME        WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
```

Notes:

- DB2 creates the default row permission on the SYSIBM.SYSCONTROLS table when you activate the permission using the ALTER TABLE command.
- DB2 creates the default row permission with the name SYS_DEFAULT_ROW_PERMISSION_*xxx*, where xxx is the table name.
- You receive SQLCODE = +100 when you are trying to execute any DML statement that is not in your answer set when the row permission is activated.
- SESSION_USER is a special register that contains the primary authorization ID of the process. The data type is VARCHAR(128). We updated the column EMPNO to match our TSO user IDs (PAOLOR1, PAOLOR2, and so on) when using the SESSION_USER register.

If multiple row permissions are defined for the table, DB2 logically combines each of them with OR operators, along with the default row permission, and consolidates them into one row access control search condition. In the previous example, the row access control search would look like Example 5-24.

Example 5-24 Row access control search

```
SELECT * FROM SPF.EMP

WHERE 1 = 0 OR
VERIFY_GROUP_FOR_USER(SESSION_USER, 'MANAGER') = 1 AND
AND WORKDEPT IN('D21', 'E21') OR
EMPNO = SESSION_USER
```

5.2.3 Row access control and catalog tables

DB2 10 introduces a new catalog table and implements changes to existing catalog tables to support row and column access control. The new SYSIBM.SYSCONTROLS table stores row permissions and column masks. In addition, the following existing catalog tables are changed:

- The SYSIBM.SYSDEPENDENCIES table reflects column mask dependencies.
- The SYSIBM.SYSOBJROLEDEP table reflects row and column mask dependencies.
- The SYSIBM.SYSTABLES table reflects row and column access control.
- The SYSIBM.SYSTRIGGERS table supports the new secure attribute.
- The SYSIBM.SYSCOLUMNS table reflects column mask existence.
- The SYSIBM.SYSROUTINES table supports the new secure attribute.
- The SYSIBM.SYSUSERAUTH table reflects the new system authority CREATESECUREAUTH.

For more information about the DB2 catalog table changes for access control, refer to *DB2 10 for z/OS SQL Reference*, SC19-2983.

5.3 Column masks

A column mask is a DB2 object that uses CASE expressions to return a masked column value in place of the column value.

If the SEPARATE_SECURITY system parameter on panel DSNTIPP1 is set to YES during installation or migration, you must have SECADM authority to create a column mask. If SEPARATE_SECURITY is set no NO, you must have the SECADM or SYSADM authority.

You can create as many masks as you want and enable them at any time. DB2 enforces these objects types only if you explicitly enable them through ALTER MASK DDL and you alter the table through ALTER TABLE DDL to activate column access control.

5.3.1 DDL for masks

The new CREATE command for masks is shown in Figure 5-4.

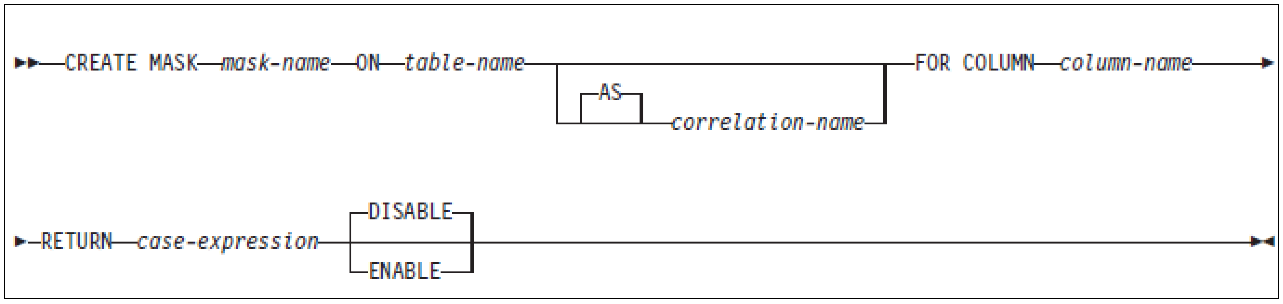


Figure 5-4 CREATE MASK SQL DDL

Where:

DISABLE	Specifies that the column mask will be disabled for column access control. The column mask remains disabled regardless of whether column access control is activated for the table.
ENABLE	Specifies that the column mask will be enabled for column access control. If column access control is not currently active for the table, the column mask becomes enabled when column access control is activated for the table. If column access control is currently active for the table, the column mask becomes enabled immediately and all packages and statements in the dynamic statement cache that reference the table are invalidated.

For details about the *case expression*, refer to *DB2 10 for z/OS SQL Reference,* SC19-2983; for examples of the case expression, refer to Chapter 10, "Implementing data access control" on page 211.

Figure 5-5 shows the ALTER command for mask column objects.

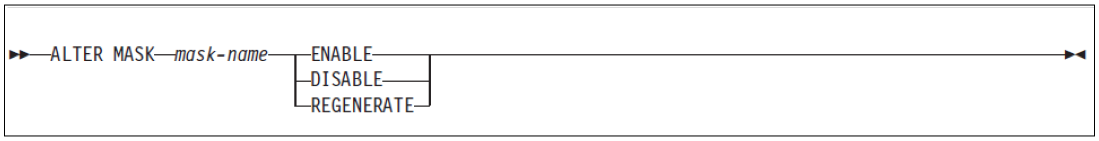


Figure 5-5 ALTER Mask DDL

The REGENERATE parameter specifies that the column mask will be regenerated.The column mask definition in the catalog is used and existing dependencies and authorizations, if any, are retained. The column mask definition is relevant as though the column mask was being created. The user-defined functions that are referenced in the column mask definition must be resolved to the same secure UDF as those that were resolved during the column mask creation.

To support the activation and deactivation of column access control, the ALTER TABLE DDL statement is extended, as shown in Figure 5-6.

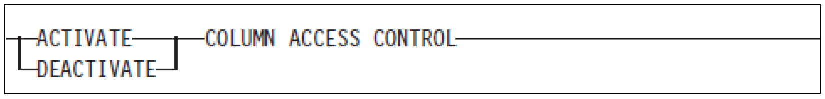


Figure 5-6 ALTER MASK column access control

If mask column access control is currently active for the table, the mask column becomes effective immediately. You can create multiple column masks per table, one for each table column. The column masks remain ineffective until you activate them. This policy does not apply to the UNLOAD and other utilities.

To check for masks, use the built-in functions described in 5.2.1, "Built-in functions" on page 90.

5.3.2 Creating and activating column masks

If the system parameter SEPARATE_SECURITY is set to YES, the SECADM authority can activate column mask access control against a table by executing one ALTER TABLE statement.

If SEPARATE_SECURITY is set to NO, you must have the SECADM or SYSADM authority.

Column mask access control can be activated even before any column mask policy is created. This situation is different from row permission access control, where, until you activate the row mask access control, no rows are returned. The number of policies to be created and enabled for column masks depends on your business rules and security goal that you need to accomplish.

We now illustrate a sample scenario. The components of the scenario are:

- Table SPF.EMP, which is created by DBA user ID PAOLOR5, who has full access to the table data.
- User ID PAOLOR2, who has SECADM authority under SECAGENT RACF group.
- User ID PAOLOR3, who can see the data but not the salary column.
- The system parameter SEPARATE_SECURITY=YES.

This scenario requires the completion of the following steps:

1. User ID PAOLOR3(CLERK) issues a SELECT command over the table, as shown in Example 5-25.

Example 5-25 CLERK selecting from the employee table

```
SELECT * FROM SPF.EMP;
---------------------------------------

EMPNO   FIRSTNME  WORKDEPT      SALARY
*       *         *                  *
------- --------- -------- -----------
PAOLOR8 CHISTINE  A00         52770.00
000020  MICHAEL   B01         41250.00
000030  SALLY     C01         38250.00
000050  JOHN      E01         40175.00
000060  IRVING    D11         32250.00
PAOLOR6 EVA       D21         36170.00
PAOLOR3 EILEEN    E11         29750.00
000100  THEODORE  E21         26150.00
```

2. User ID PAOLOR2(SECADM) issues a CREATE column MASK command and activates it, as shown in Example 5-26.

Example 5-26 Column MASK object creation using RACF group

```
CREATE MASK SALARY_COLUMN_MASK ON
   FOR COLUMN SALARY RETURN
     CASE WHEN(VERIFY_GROUP_FOR_USER(SESSION_USER,'PROJL')=1  OR
               VERIFY_GROUP_FOR_USER(SESSION_USER,'CLERK')=1)
         THEN NULL
         ELSE SALARY
     END
ENABLE;

ALTER TABLE SPF.EMP ACTIVATE COLUMN ACCESS CONTROL;
```

3. User ID PAOLOR3(CLERK) issues a SELECT command over the table, as shown in Example 5-27.

Example 5-27 Results from SELECT for user CLERK after mask

```
SELECT * FROM SPF.EMP;
---------------------------------------

EMPNO   FIRSTNAME  WORKDEPT      SALARY
*       *         *                  *
------- --------- -------- -----------
PAOLOR8 CHISTINE  A00                ?
000020  MICHAEL   B01                ?
000030  SALLY     C01                ?
000050  JOHN      E01                ?
000060  IRVING    D11                ?
PAOLOR6 EVA       D21                ?
PAOLOR3 EILEEN    E11                ?
000100  THEODORE  E21                ?
```

Rule applicability: During table access, DB2 applies these rules to every user, including the table owner and the SYSADM, SECADM, SYSTEM DBADM, and DBADM authorities.

As for RACF groups, you can create masks to check against role and trusted context. As shown in Example 5-28, you have to change the built-in function of VERIFY_ROLE_FOR_USER. The roles and trusted context should be created to satisfy this mask.

Example 5-28 Column MASK object creation using role and trusted context

```
CREATE MASK SALARY_COLUMN_MASK ON DSN81010.EMP
   FOR COLUMN SALARY RETURN
     CASE WHEN(VERIFY_ROLE_FOR_USER(SESSION_USER,'SALES_ROLE')=1)
         THEN CASE WHEN(SALARY > 50000) THEN 1.0
                   WHEN(SALARY > 30000) THEN 2.0
                   WHEN(SALARY > 16000) THEN 3.0
                   ELSE 0.0
              END
         ELSE
         CASE WHEN(VERIFY_ROLE_FOR_USER(SESSION_USER,'HR_ROLE')=1)
              THEN SALARY
              ELSE NULL
         END
     END
ENABLE;
```

In Example 5-29, we show how you can mask column dates. User IDs for the SLITSO RACF group had, as output, the year changed to 9999; for RACF group HRITSO, the output is the real date and for the others the output is the current date.

Example 5-29 Masking dates

```
CREATE MASK SALARY_COLUMN_HIDE ON DSN81010.EMP
   FOR COLUMN HIREDATE RETURN
     CASE WHEN(VERIFY_GROUP_FOR_USER(SESSION_USER,'SLITSO')=1)
         THEN  '9999'!!'-'!!DAY(HIREDATE)!!'-'!!MONTH(HIREDATE)
         ELSE
            CASE WHEN(VERIFY_GROUP_FOR_USER(SESSION_USER,'HRITSO')=1)
                THEN HIREDATE
                ELSE CURRENT DATE
            END
     END
ENABLE;
```

5.4 EXPLAIN table information

When you explain an SQL statement that references a row or column access control enforced table, the plan table shows any table that is referenced by that query at the time the optimizer selected the access path for that SQL statement. This information includes references to tables that are referenced in row permission or column mask objects and not shown in the explained SQL statement. This action allows for row permission predicates and column masks to be included in query analysis and visualization by tools.

Using the table SPF.EMP, we can show some changes for explain tables in DB2 10. In Example 5-30, we create the row permission and execute the SQL statement to explain a query that selects all rows for that table.

Example 5-30 Row permission and explain query

```
CREATE PERMISSION SECHEAD.ROW_EMP_RULES ON SPF.EMP
  FOR ROWS
  WHERE (VERIFY_GROUP_FOR_USER(SESSION_USER, 'CLERK') = 1
     OR VERIFY_GROUP_FOR_USER(SESSION_USER, 'DIRECTOR') = 1
     OR VERIFY_GROUP_FOR_USER(SESSION_USER, 'PROJL') = 1
     OR VERIFY_GROUP_FOR_USER(SESSION_USER, 'SUPERV') = 1
     OR VERIFY_GROUP_FOR_USER(SESSION_USER, 'MANAGER') = 1
    AND WORKDEPT IN('D21', 'E21')
     OR EMPNO = SESSION_USER)
  ENFORCED FOR ALL ACCESS
  ENABLE ;

EXPLAIN ALL SET QUERYNO=123 FOR
 SELECT * FROM SPF.EMP;
```

5.4.1 DSN_PREDICAT_TABLE information

DB210 introduces new columns to explain tables such as DSN_PREDICAT_TABLE to support additional information about row permission filters and column mask information in SQL explain.

Table 5-1 shows the new columns.

Table 5-1 DSN_PREDICAT_TABLE columns

Column name	Format	Description
ORIGIN	CHAR(1) NOT NULL WITH DEFAULT	The predicates originate from: ▶ Blank - DB2 generated ▶ C - Column mask ▶ R - Row permission ▶ U - user specified
CLAUSE	CHAR(8) NOT NULL	The clause where the predicate exists is SELECT - select clause.

The extract of the query output in Example 5-31 shows the predicates DB2 injected into the query for the row permission that exists on that table. This information is important if you have to reconstruct the full predicate, including those injected by DB2 because of row permission and column mask objects.

Example 5-31 DSN_PREDICAT_TABLE query output

```
SELECT CLAUSE,ORIGIN,TEXT FROM DB2R4.DSN_PREDICAT_TABLE
WHERE ORIGIN IN('C','R') AND QUERYNO = 123;

CLAUSE   ORIGIN TEXT
*        *      *
-------- ------ ------------------------------------------------------------
WHERE    R      (((((VERIFY_GROUP_FOR_USER(USER,'CLERK')=1 OR VERIFY_GROUP_FO
WHERE    R      VERIFY_GROUP_FOR_USER(USER,'CLERK')=1
WHERE    R      VERIFY_GROUP_FOR_USER(USER,'DIRECTOR')=1
WHERE    R      VERIFY_GROUP_FOR_USER(USER,'PROJL')=1
WHERE    R      VERIFY_GROUP_FOR_USER(USER,'SUPERV')=1
WHERE    R      (VERIFY_GROUP_FOR_USER(USER,'MANAGER')=1 AND SPF.EMP.WORKDEPT
WHERE    R      VERIFY_GROUP_FOR_USER(USER,'MANAGER')=1
WHERE    R      SPF.EMP.WORKDEPT IN ('D21','E21')
WHERE    R      SPF.EMP.EMPNO=USER
```

5.4.2 DSN_STRUCT_TABLE information

DB210 introduces a new column to explain tables, such as DSN_STRUC_TABLE, to support additional information about row permission filters and column mask information in SQL explain.

Table 5-2 shows the new column.

Table 5-2 DSN_STRUC_TABLE

Column name	Format	Description
ORIGIN	CHAR(01) NOT NULL WITH DEFAULT	The query block originates from: ▶ Blank - DB2 generated ▶ C - Column mask ▶ R - Row permission ▶ U - user specified

DB2 externalizes additional information related to row permission and column mask objects into DSN_STRUC_TABLE that provides information, such as number of rows returned by the data manager (DM) and the relational data system (RDS) component at query block level, which is typically used for detailed access path analysis.

Example 5-32 shows an extract of DSN_STRUC_TABLE.

Example 5-32 DSN_STRUC_TABLE query output

```
SELECT TIMES, ROWCOUNT, ORIGIN  FROM DB2R4.DSN_STRUCT_TABLE
WHERE ORIGIN IN('C','R','U') AND QUERYNO = 123;

TIMES                    ROWCOUNT ORIGIN
                     *          * *
--------------------- ----------- ------
 0.000000000000000E+00          0 U
 1.000000000000000E+04      10000 U
 0.000000000000000E+00          0 U
 1.000000000000000E+04       2786 U
 0.000000000000000E+00          0 U
 1.000000000000000E+04      10000 U
 0.000000000000000E+00          0 U
 1.000000000000000E+04       2172 U
```

These tables are created by the DSNTESC member on the SDSNSAMP data set.

5.5 Triggers and UDF information

To create a trigger for a table that is enforced with row and access control, you must have the new system privilege CREATE_SECURE_OBJECT, which is required to secure triggers and scalar functions. If you use triggers or functions on tables that are enforced by row permissions or column masks, your trigger or function needs to be marked as secure. The attribute is SECURED. If a trigger exists but is not secure, row or column access control cannot be activated for the associated table.The SECADM user ID should issue `GRANT CREATE_SECURE_OBJECT to *xxx*`, where xxx could be one RACF group ID. After the trigger has been created, the SECADM should revoke this privilege from xxx.

By default, DB2 marks a UDF as insecure for row and column access control. If the argument of a UDF references a column mask enforced column, the UDF is always required to be secure, regardless of whether the UDF is referenced in a SELECT list for output or in a predicate.

Table 5-3 shows the tables containing the SECURE column.

Table 5-3 .Tables with secure column information

Column name	Table name	Format	Description
SECURE	SYSTRIGGERS	CHAR(01) NOT NULL	The following settings determine the trigger's security: ► Y: Secure ► N: Not secure
SECURE	DSN_FUNCTION_TABLE	CHAR(01) NOT NULL	The following settings determine the UDF's security: ► Y: Secure ► N: Not secure
SECURE	SYSROUTINES	CHAR(01)	The following settings determine the routine's security: ► Y: Secure ► N: Not secure

6

Cryptography for DB2 data

In this chapter, we describe the various ways in which DB2 for z/OS can use cryptographic features of the System z hardware family, z/OS operating system cryptographic services, and control unit based encryption. In addition, we briefly discuss the use of network based encryption to protect remote client data flows into and from the DB2 for z/OS server.

We also describe the main functions for protecting DB2 data at rest: disk and tape encryption. Tapes are archived, disk drives are replaced routinely, and entire systems are retired. When drives or tape cartridges are physically removed from the storage system, IBM self-encrypting storage solutions automatically protect the data from any system not authorized to read the data. These drives are designed to encrypt data automatically as it enters the drive to be stored, and then automatically decrypt it as it moves out of the drive. The embedded encryption engine helps ensure that there is virtually no performance degradation compared to non-encrypting drives.

This chapter contains the following topics:

- DB2 built-in-function support for encryption
- InfoSphere Guardium Data Encryption for DB2 and IMS Databases
- Disk storage based encryption with IBM System Storage DS8000
- Tape storage encryption
- Overview of SSL and IP AT-TLS

6.1 DB2 built-in-function support for encryption

Starting with Version 8, DB2 for z/OS provides a number of built-in functions that allow you to encrypt data at the column level. These functions include ENCRYPT_TDES (or ENCRYPT) to encrypt data in a column, and DECRYPT_BIN and DECRYPT_CHAR to decrypt the data in its appropriate format, and the GETHINT function to retrieve the hint for the password.

6.1.1 Insert and create

The SET ENCRYPTION PASSWORD statement allows you to specify a password as a key to encryption. Example 6-1 shows an example using ENCRYPT to insert rows.

Example 6-1 SQL insert example using ENCRYPT

```
CREATE TABLE EMPL
(EMPNO VARCHAR(64) FOR BIT DATA,
EMPNAME CHAR(20),
CITY CHAR(20) NOT NULL DEFAULT 'KANSAS CITY',
SALARY DECIMAL(9,2))
IN DSNDB04.RAMATEST ;
COMMIT ;
SET ENCRYPTION PASSWORD = 'PEEKAY' WITH HINT 'ROTTIE' ;
INSERT INTO EMPL(EMPNO,EMPNAME, SALARY)
VALUES (ENCRYPT('12346'),'PAOLO BRUNI',20000.00) ;
INSERT INTO EMPL(EMPNO,EMPNAME, SALARY)
VALUES (ENCRYPT('12347'),'ERNIE MANCILL',20000.00) ;
```

When creating a column for data encryption, you must define it as VARCHAR. The length of the VARCHAR depends on the password and the password hint. Assuming EMPNO is VARCHAR(6) before encryption, you can compute the final length of VARCHAR, as shown in Example 6-1.

Table 6-1 Sample column encrypted VARCHAR length

Description	**Length in bytes**
Maximum length of un-encrypted data	6
Number of bytes rounded up to 8-byte boundary	2
Encrypted password phrase	24
Optional password hint	32
Total	**64**

Therefore, if you do not use a password hint, define the column for encrypted data as VARCHAR(32) FOR BIT DATA. If you use a password hint, DB2 requires an additional 32 bytes to store the hint. In this case, you must define the EMPNO column as VARCHAR(64) FOR BIT DATA.

The application is responsible for managing all these keys. Make sure you have a mechanism in place to manage and protect the passwords that are used to encrypt the data. Use the password hint. The GETHINT function returns the password hint for every row in the table. Without the password, there is no way to decrypt the data.

These encryption functions use the Triple Data Encryption Standard (DES) to perform the encryption. Encryption is performed as follows for an insert operation:

- The application supplied password is hashed using the MD-5 function. The MD-5 Message-Digest Algorithm is a widely used cryptographic hash function with a 128-bit (16-byte) hash value. Specified in RFC 1321 Network Working Group, MD-5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. An MD-5 hash is typically expressed as a 32-digit hexadecimal number. The important characteristic of MD-5 hashing is that when used to hash a value repeatedly, the same hash value is returned.
- The resulting 128-bit value is then used as the data encrypting key to encrypt the password along with the clear-text column value. TDES clear key encryption using CPACF is used for the encryption algorithm.

6.1.2 SELECT

To retrieve the data, the DECRYPT_CHAR function must be applied to EMPNO, as shown in Example 6-2.

Example 6-2 Select SQL example using DECRYPT

```
SET ENCRYPTION PASSWORD = 'PEEKAY' ;
SELECT SUBSTR(DECRYPT_CHAR(EMPNO),1,6) AS EMPNO,
EMPNAME,CITY,SALARY
FROM EMPL ;
```

Decryption is performed during select processing as follows:

- The application supplied password is hashed using the previously described MD-5 function. The resulting MD-5 128-bit value is then used to encrypt the application supplied password using TDES clear key processing.
- The encrypted application supplied password is then compared to the encrypted password value stored in the encrypted table column value.
- If the password matches, the column value is decrypted using the same derived 128-bit value, and the clear-text value is presented back to the SQL application requestor.
- If the password does not match, the application is passed back the encrypted ciphertext column value with no indication of a password mismatch. It is up to the application to confirm successful or unsuccessful decryption has been performed.

6.1.3 Prerequisites and considerations

To use the DB2 built-in-functions for encryption, the following prerequisites are needed:

- DB2 for z/OS Version 8 or later
- A functional ICSF environment on the LPAR hosting the DB2 subsystem
- On System z, IBM eServer™ zSeries® 890 and 990, CPACF needs to be available and enabled
- On zSeries 800, 900, or earlier, PCICC or Cryptographic Coprocessor Feature (CCF)

ICSF needs to be functional to use the MD-5 hash callable service. CPACF and ICSF need to be available to perform clear key TDES encryption operations.

Each CP on the z990 and later models has an assist processor on the chip in support of cryptography. This feature provides for hardware encryption and decryption support. PCIXCC provides a cryptographic environment with additional functions. To learn more about PCIXCC, refer to *IBM eServer zSeries 990 (z990) Cryptography Implementation*, SG24-7070.

Existing applications that need to implement DB2 encryption need to be changed to apply the DB2 encrypt and decrypt built-in functions to each column to be encrypted or decrypted. All encrypted columns must be declared for bit data. Unchanged read-applications see data in encrypted form. Applications may supply a different key for each column, but may also supply the key in a special register. We suggest, for less impact on performance, that you specify the key in the special register.

If data replication to another system is active, definitions of columns and the encryption key must be created using the same source key and identity value.

The LOAD and UNLOAD utilities do not support the DB2 built-in encryption functions, but do handle broader encryption. SQL-based programs such as DSNTIAUL support encryption. Encryption of numeric fields is not supported. The length of encrypted columns must allow for an additional 24 bytes, rounded up to a double-word boundary, for storing the encryption key. Space usage may be a concern if you plan to use DB2 to encrypt small columns.

Indexes are also encrypted. Because of this, predicates that depend on the collating sequence of encrypted columns (for example, range predicates) may produce the wrong results (unless modified to use built-in functions correctly).

For example, the following statement produces the wrong results:

```
SELECT COUNT(*) WHERE COL =:HV;
```

The following statement produces the correct results with almost no impact to performance:

```
SELECT COUNT(*) WHERE COL = ENCRYPT_TDES(:HV);
```

The following statement produces the wrong results:

```
SELECT COUNT(*) WHERE COL < ENCRYPT_TDES(:HV);
```

The following statement produces the correct results with a large impact on performance:

```
SELECT COUNT(*) WHERE DECRYPT_CHAR(COL) <:HV;
```

6.2 InfoSphere Guardium Data Encryption for DB2 and IMS Databases

InfoSphere Guardium Data Encryption Tool for DB2 and IMS Databases is the tool that provides you with a data encryption function for both DB2 and IMS for z/OS databases in a single product. It enables you to protect your sensitive and private data for IMS at the segment level and for DB2 at the table level.

This tool performs encryption using EDITPROCs on the full row. The restrictions due to the use of EDITPROC apply. The most notable are that indexes are not encrypted and the ALTER is not available.

Unlike the DB2 encryption functions shipped with DB2, the Data Encryption Tool uses different keys to encrypt different tables. The tools supports encryption based on clear keys, secure keys, or Protected Key CPACF. All keys are managed through ICSF key management services, and no application management of passwords or keys are required. Keys are stored in the ICSF cryptographic key data set (CKDS), and are created and administered by the ICSF key administrator. Clear key or Protected Key CPACF generally perform better than secure key encryption. The tool supports single, double, or triple DES keys, as well as AES keys up to 256 bits in length. Refer to *IBM eServer zSeries 990 (z990) Cryptography Implementation*, SG24-7070 and *IBM System z10 Enterprise Class Technical Guide*, SG24-7516 to learn more about the clear and secure keys.

Customization consists of the following tasks:

- Building a user exit routine (DB2 EDITPROC exit)
- Putting the user-specified encryption key label in the exit routine

The tool provides sample jobs to help you build the user exit (DB2 EDITPROC) and specify the encryption key label. Alternatively, ISPF dialogs, supplied by the tool, can be used to build the exit.

InfoSphere Guardium Data Encryption for DB2 and IMS Databases supports all versions of DB2. It encrypts only the whole data row. No application changes are required. However, you can include the EDITPROC only at the CREATE time of a table. DROP, CREATE, RUNSTATS, and BIND are necessary for adding the EDITPROC to an existing table. Once implemented with EDITPROC, encryption is transparent, and applications do not need to be aware of encryption keys.

The tool takes care of data encryption on disk (data at rest). The data on channel, the buffer pools, the image copies, and the logs are encrypted. Data passed to applications and the indexes are not encrypted. This means that existing authorization controls are unaffected. After the data is brought, for example, into DBMS working storage, you are using the existing DB2 and IMS authorizations to secure data.

When installed, an EDITPROC generated by the InfoSphere Guardium Data Encryption for DB2 and IMS Databases performs without any requirements for application changes.

Figure 6-1 shows a representation of insert processing with the EDITPROC.

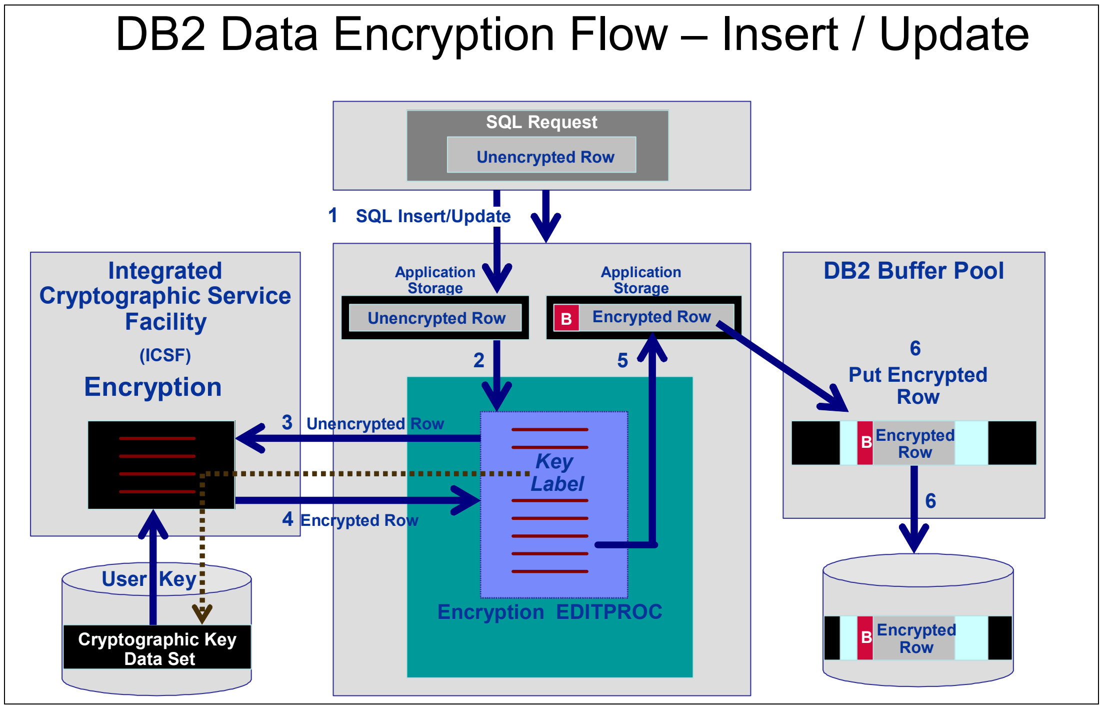


Figure 6-1 Encryption flow with EDITPROC an INSERT example

Insert processing occurs as follows:

1. The application requestor issues the SQL INSERT request and is passed to DB2.
2. DB2 determines an EDITPROC is defined for the table, and passes control to the EDITPROC.
3. The EDITPROC passes the embedded ICSF keylabel to ICSF, which locates the key in the CKDS. If this is a clear key, the data is then encrypted using the KMC instruction. If the key is a secure key, the encryption request is handled through an API call with interaction with either the CEX2C/CEX3C or through Protected Key CPACF processing.
4. The encrypted row value is passed back to the EDITPROC by ICSF services.
5. At this point, the row value shown in the application storage is now represented in ciphertext.
6. The encrypted row value is placed into the DB2 buffer pool page, and at a point determined by DB2, is externalized to the VSAM linear data set. The VSAM control interval record contains encrypted data.

One characteristic of this encryption flow is that on insert and update processing, as the row image is written to the DB2 recovery log, the log record contains encrypted data. So, when archived recovery log assets are shipped offsite, to support disaster recovery efforts, for example, these assets are protected because the log data is encrypted.

In addition, because the linear VSAM data sets contain encrypted pages, when the DB2 COPY utility is used to create backups for recovery, these Image Copy data sets also contain encrypted data and are also protected.

6.2.1 Deciding between EDITPROC, DB2 based encryption, and disk encryption

Different considerations can drive customers towards one approach versus another when evaluating an implementation of encrypting DB2 data at rest. Table 6-2 summarizes the different characteristics of the DB2 built-in-function, the disk encryption, and the InfoSphere Guardium Data Encryption for DB2 and IMS Databases.

Table 6-2 Comparison among DB2 BIF, Data Encryption for DB2 and IMS Databases and disk encryption

Element	**Column (DB2 built in functions)**	**Row/Table (InfoSphere Guardium Data Encryption for DB2 and IMS Databases)**	**Disk level encryption**
DB2 support	► Version 8, 9, and 10 ► Data in indexes is encrypted. ► Does not work with DB2 Unload/Load Utility. ► Data type of encrypted columns must be FOR BIT DATA. ► Problematic access paths due to index encryption.	► Version 7.x, 8.x, 9.x, and 10.x. ► DB2 index data is not encrypted. ► Works with all DB2 utilities.	Transparent.
Application change required	Application must change to invoke the BIFs for the columns that will be encrypted.	No application change, but each table will need to be recreated with an EDITPROC.	No.
Transaction processing impact	The cost impact depends on hardware, DB2, and application access.	The cost impact depends on hardware, DB2, and application access.	Negligible.
Key management	The application has responsibility for the encryption key.	Keys are managed by and accessed through ICSF.	Key management services are supported by IBM Tivoli Key Lifecycle Manager software (TKLM).
Prerequisites	► ICSF must be active. ► CPACF hardware.	► ICSF must be active. ► Secure PCI card, unless running HCR7751 or later and clear key only CKDS.	Encrypting is at the ES8000 level. For each ES8000, there is a single factory defined key.

Encryption, as implemented with the DB2 built-in-function, requires application changes to invoke the SQL encryption extensions, and being able to securely manage the required password within the application is problematic. In addition, the inability of LOAD and UNLOAD to be able to consume the password creates operational difficulties that typically requires the development of application programs to mimic the function of UNLOAD and LOAD. Retrofitting DB2 encryption based on the built-in-function can be difficult and typically it is easier to implement when designing and implementing new applications. Key rotation is another problematic area and, depending on the way that password management has been implemented, key rotation can require additional application changes. However, the DB2 built-in-function encrypts indexes, and for some customer requirements, this is viewed as a requirement for regulatory or industry compliance.

InfoSphere Guardium Data Encryption for DB2 and IMS Databases does not require application changes to be implemented. Although there is an additional impact associated with encryption as implemented by Data Encryption Tool for IMS and DB2 Databases, because indexes are not encrypted, there are no problematic access path issues with which to contend. In addition, DB2 and all popular third-party tools and utilities coexist with tool based encryption. Key rotation, particularly when a Protected Key CPACF approach is taken, can be performed with little application or administrative impact. However, as mentioned above, indexes are not encrypted, and for some customers, this represents a security exposure. In some situations, indexes can be redesigned to minimize this exposure.

6.3 Disk storage based encryption with IBM System Storage DS8000

IBM recognizes the need for data protection, not only from hardware or software failures, but also from physical relocation of hardware, theft, and re-tasking of existing hardware. Full Disk Encryption (FDE) drive sets provide the ability to encrypt data at rest on an IBM System Storage® DS8000® series storage controller, helping to mitigate the threat of theft, mismanagement, or loss of business critical data. The DS8000 series has the capability to allow customers to install encrypted 146 GB 15,000 rpm, 300 GB 15,000 rpm, and 450 GB 15,000 rpm Fibre Channel drives with key management services supported by Tivoli Key Lifecycle Manager software (TKLM) (see “Tivoli Key Lifecycle Manager” on page 404). The Full Disk Encryption disk drive sets are optional to the DS8000 series.

6.3.1 DS8000 data encryption management overview

The Full Disk Encryption disk drive function of DS800 is shown in Figure 6-2.

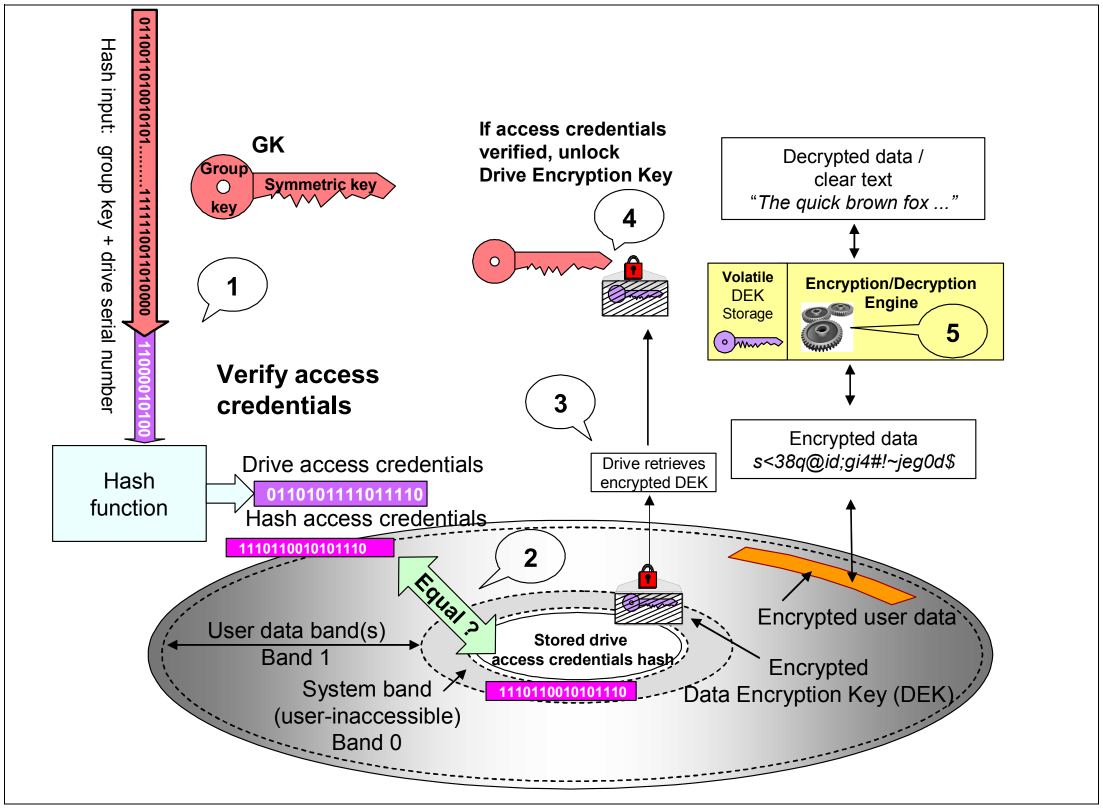

Hash input: group key + drive serial number
01100110100101.........1111100101010000
1100001010
GK
Group key
Symmetric key
1
Verify access credentials
Hash function
Drive access credentials
0110101111011110
Hash access credentials
1110110010101110
Equal ?
2
If access credentials verified, unlock Drive Encryption Key
4
3
Drive retrieves encrypted DEK
Decrypted data / clear text "The quick brown fox ..."
Volatile DEK Storage
Encryption/Decryption Engine
5
Encrypted data s<38q@id;gi4#!~jeg0d$
Encrypted user data
Encrypted Data Encryption Key (DEK)
Stored drive access credentials hash
1110110010101110
User data band(s) Band 1
System band (user-inaccessible) Band 0

Figure 6-2 DS8000 Storage Encryption

Using the TKLM, the customer configures one or more storage facility images by completing the following steps:

1. One or more keys (and associated key labels) are defined within the TKLM.
2. One or more TKLM IP ports are defined for each DS8000, up to four separate ports are supported, and a minimum of two TKLM ports are suggested for redundancy.
3. One or more encryption groups are configured on the DS8000. The encryption group configuration includes an associated key label. After the encryption group is configured, the TKLM is contacted, passing the key label. The TKLM performs a key validation and exchange with the DS8000 and passes back one or more keys based on the number of key labels exchanged. This exchange is managed by another component called the Isolated Key Server (SFI). The link between the TKLM and the SFI is through secured SSL.
4. Using the TKLM key(s), the band encryption key, which is set at the factory, is wrapped.
5. The customer then configures an encryption group, which have one or more encryption capable ranks in an extent pool.
6. The DS8000 locks data bands that are located on encrypting disks in extent pools that have been configured into an encryption group.
7. The customer then configures logical volumes in the extent pool. Customer data for these logical volumes are stored on disks with locked data bands.

8. If a rank is removed, the disks on the ranks have their wrapped encryption key reset. This action also causes all of the data stored on the disks to be erased, which is referred to as a *cryptographic erasure*.

The Full Disk Encryption support feature is available only as a plant order. Plant-configured encryption supporting systems is allowed to increase the number of drive sets installed at the installed location. Intermixing of drives is not supported, so the entire subsystem is either encrypted drives or intermixed devices of Fibre Channel, SATA, and SSD devices.

6.3.2 Disk encryption details

Each FDE drive has an encryption key for the area of the drive that contains customer data. When the customer data area is unlocked, the FDE drive still encrypts and decrypts the data with a data encryption key and this data encryption key is also wrapped (encrypted) with access credentials. Here, a default encryption key is used to encrypt the data encryption key, but it is done transparently to the initiator (DS8700). If someone, however, takes the disk plate without the interface, and somehow tries to read from the disks, it would be impossible because the data is encrypted.

An FDE drive that is made a member of an encryption-enabled rank is locked. The FDE drive is unlocked when it is unassigned, is a spare, or is a member of an encryption-disabled rank. Locking occurs when an FDE drive is added to an encryption-enabled rank either during rank creation or sparing. Unlocking occurs when an encryption-enabled rank is deleted or a member of an encryption-enabled rank is reused as a spare. Unlocking always results in a cryptographic erasure of a FDE drive (the disk resets its own encryption key). This also happens when an encryption-disabled rank is deleted.

In a cryptographic erasure, a new data encryption key is generated in each disk drive. The new key is encrypted with default access credentials, and both the access credentials and the encrypted data encryption key are stored on band 0 of the drive. Now the drive is unlocked. If someone were to try to read the old data, nonsense data would be returned because decryption now uses another key, which does not decrypt the data any longer.

The data encryption key for the data area is wrapped (encrypted) with an access credential produced with the group key. This access credential is converted to a secure hash and stored on the disk. At that stage, the customer data area is locked.

After a disk power loss or power off, the read/write access to the data on a locked area is blocked until the DS8000 has authenticated by supplying the currently active access credential, that is, the group key. (The DS8700 must first get unlock keys for the group key from the Tivoli Key Lifecycle Manager server). The following steps occur:

1. The disk drives verifies the access credentials (containing the group key).
2. The drive validates the access credentials with the one stored on the disk drive.
3. The drive reads the stored encrypted data key.
4. The encrypted data key is decrypted by using access credentials (group key).

6.3.3 Characteristics of the ES8000 encryption implementation

As currently implemented, there is no mixture of encrypting and non-encrypting disks in an ES8000. For each ES8000, there is a single factory defined key, which is wrapped at the local site with the TKLM managed wrapper. Thus, all disks on the single ES8000 share the same key.

After the ES800 powers up and connects to the TKLM, there is a key exchange using the keylabel, and if the credentials match, then any subsequent access (I/O) to the subsystem retrieves clear text data.

So, although there is clearly demonstrated protection of the data at rest from attack vectors such as removal of physical media by the CE, or retirement or return of older subsystems, there are other avenues of attack that can justify additional layers of encryption protection. The ES8000 implementation should be viewed as just one element in a layered environment where other encryption solutions can complement the ES8000.

One scenario to describe the requirement for additional defenses is one where the storage administrator has rights to access data at a volume level, but might not have underlying access to the DB2 linear data sets. Because there are catalog privileges granted, the use of tools such as DFSMSdss could allow access to the underlying data, outside of any SQL-based access. A second example would be where a z/OS system programmer brings up another LPAR with little or no RACF protection active. The volumes would be varied offline from the production system, and then online to the sandbox environment, and data could then be compromised in any number of ways without the use of SQL.

In the above scenarios, the ES8000 encryption would provide no protection from nefarious access, but additional encryption techniques, such as those provided by the use of the InfoSphere Guardium Data Encryption Tool for IMS and DB2 Databases, would further protect the data by encryption implemented at another layer.

6.4 Tape storage encryption

The IBM System Storage TS1120 Tape Drive was the first drive to offer the option of using drive-based data encryption. IBM now has three tape drives that are capable of encrypting the data written on the tape cartridge. They are the IBM System Storage TS1120 Model E05 Tape Drive, IBM System Storage TS1130 Model E06 and Model EU6 Tape Drive, and the IBM Systems Storage Linear Tape-Open (LTO) Ultrium Generation 4 Tape Drive. In this section, we refer to them as the TS1120 tape drive, TS1130 tape drive, and the LTO-4 tape drive.

While other encryption solutions require hardware resources or processor power when using software encryption, tape encryption is done with little or no impact on the performance of the TS1120 tape drive or the LTO4 tape drive. You can easily exchange encrypted tapes with your data centers or other parties that have the necessary key information to decrypt the data.

With the original encryption announcement for the TS1120 tape drive, IBM also introduced a new IBM Encryption Key Manager component for the Java Platform feature (Encryption Key Manager (EKM)) that is designed to generate and communicate encryption keys for tape drives across the enterprise. The new feature uses standard key repositories on supported platforms. The IBM Tape Encryption solution provides an enterprise key management solution with common software for open systems and mainframe environments that allow sharing of a common keystore across platforms. Integration with z/OS policy, key management, and security capabilities provides a proven, highly secure infrastructure for encryption key management.

IBM tape encryption provides high performance data encryption. Encryption is performed at the tape drive hardware at a speed of up to 120 Mbps (for un-compressed data) and even higher speeds for data that compresses. It also supports encryption of large amounts of tape data for backup and archive purposes.

The IBM Tape Encryption solution, using the TS1120 Tape Drive or the LTO-4 Tape Drive, offers a cost-effective solution for tape data encryption by offloading encryption tasks from the servers, using the existing tape infrastructure incorporated in standard IBM Tape Libraries, and eliminating the need for unique appliance hardware.

6.4.1 How tape encryption works

Encryption, implemented in the tape drive, encrypts the data before it is written to the cartridge. When tape compression is enabled, the tape drive first compresses the data to be written and then encrypts it., which means that there is no loss of capacity with IBM Tape Encryption. If the encryption solution encrypts the data first and then tries to compress the data, the encrypted data usually has little or no compression.

To encrypt the data, the tape drive needs a key. The key is provided by the Encryption Key Manager, and it is provided in an encrypted form to make the tape encryption solution secure. Key management can be handled either internally by an application, such as Tivoli Storage Manager, or externally by an Encryption Key Manager.

The IBM Encryption Key Manager (EKM) component for the Java platform is a Java software program that assists IBM encryption-enabled TS1120 tape drives and Linear Tape-Open (LTO) Ultrium 4 tape drives by providing, protecting, storing, and maintaining encryption keys that are used to encrypt information being written to, and decrypt information being read from, tape media. EKM operates on a variety of operating systems. EKM is designed to be a shared resource deployed in several locations within an enterprise. It is capable of serving numerous IBM encrypting tape drives regardless of where those drives reside (for example, in tape library subsystems, connected to mainframe systems through various types of channel connections, or installed in other computing systems). IBM supplies the EKM at no additional cost.

EKM acts as a process awaiting key generation or key retrieval requests sent to it through a TCP/IP communication path between EKM and the tape library, tape controller, tape subsystem, device driver, or tape drive. When a tape drive writes encrypted data, it first requests an encryption key from EKM. The tasks that the EKM performs upon receipt of the request are different for the TS1120 tape drive and the TS1040 tape drive.

With the introduction of TKLM, IBM has made available the next generation of key manager software to enable serving keys to the encrypting tape drives. TKLM is intended to give a consistent look and feel for key management tasks across the brand, while simplifying those same key management tasks.

Figure 6-3 summarizes the process flow for tape encryption using TS1120.

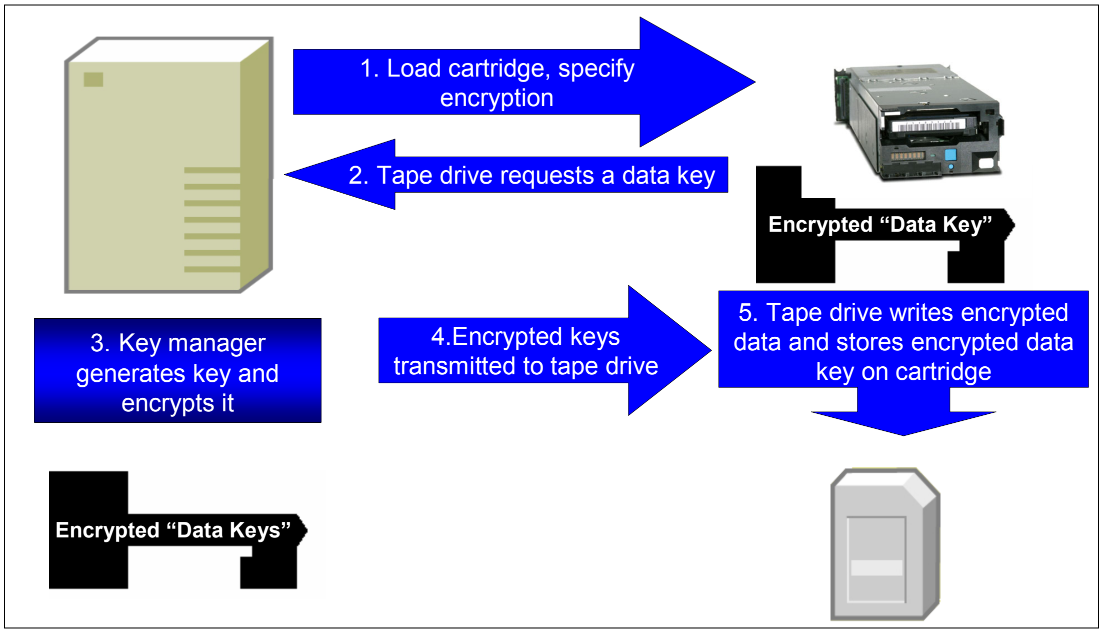

1. Load cartridge, specify encryption
2. Tape drive requests a data key
Encrypted "Data Key"
3. Key manager generates key and encrypts it
4.Encrypted keys transmitted to tape drive
5. Tape drive writes encrypted data and stores encrypted data key on cartridge
Encrypted "Data Keys"

Figure 6-3 TS1120 tape encryption process flow

EKM requests an Advanced Encryption Standard (AES) key from the cryptographic services and serves it to the tape drives in two protected forms:

- Encrypted or wrapped, using Rivest-Shamir-Adleman (RSA) key pairs. TS1120 tape drives write this copy of the key to the cartridge memory and three additional places on the tape media in the cartridge for redundancy.
- Separately wrapped for secure transfer to the tape drive, where it is unwrapped upon arrival and the key inside is used to encrypt the data being written to tape.

When an encrypted tape cartridge is read by a TS1120 tape drive, the protected AES key on the tape is sent to EKM where the wrapped AES key is unwrapped. The AES key is then wrapped with a different key for secure transfer back to the tape drive, where it is unwrapped and used to decrypt the data stored on the tape. EKM also allows protected AES keys to be re-wrapped, or re-keyed, using different RSA keys from the original keys that were used when the tape was written. Re-keying is useful when an unexpected need arises to export volumes where the public keys were not included. It eliminates the need to rewrite the entire tape and enables a tape cartridge's data key to be re-encrypted with a public key.

Figure 6-4 summarizes the LTO4 tape encryption process flow.

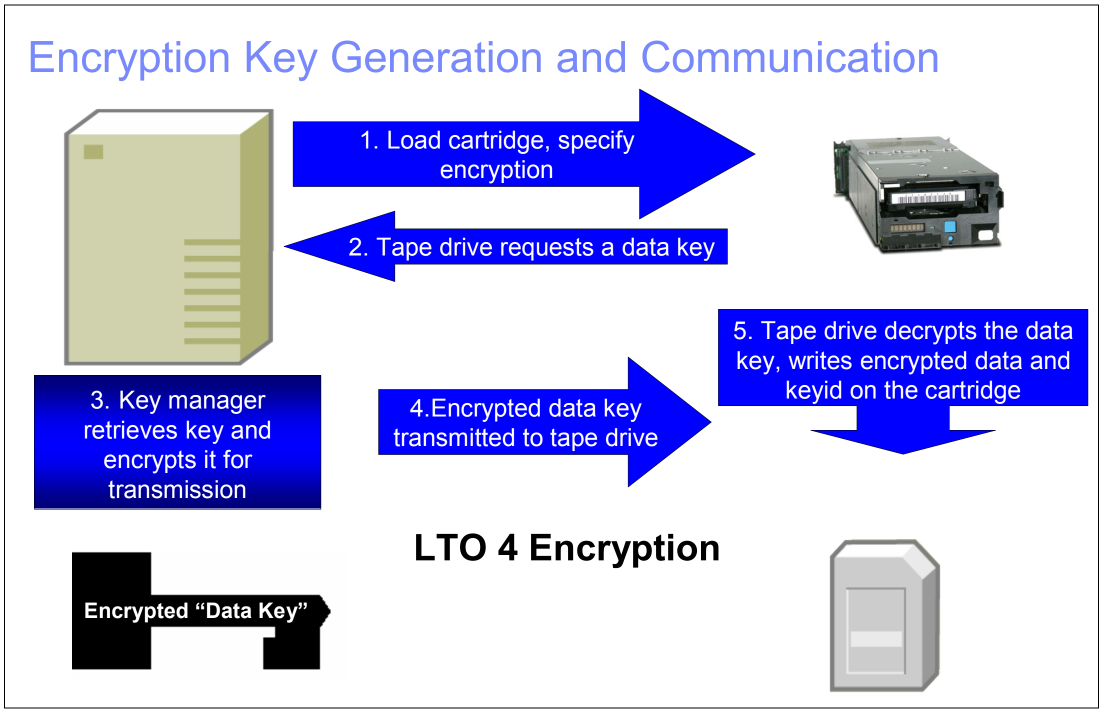

Encryption Key Generation and Communication
1. Load cartridge, specify encryption
2. Tape drive requests a data key
3. Key manager retrieves key and encrypts it for transmission
4.Encrypted data key transmitted to tape drive
5. Tape drive decrypts the data key, writes encrypted data and keyid on the cartridge
LTO 4 Encryption
Encrypted "Data Key"

Figure 6-4 LTO4 tape encryption process

The EKM fetches an existing AES key from a keystore and wraps it for secure transfer to the tape drive where it is unwrapped upon arrival and used to encrypt the data being written to tape.

When an encrypted tape is read by an LTO Ultrium 4 tape drive, the EKM fetches the required key from the keystore, based on the information in the Key ID on the tape, and serves it to the tape drive wrapped for secure transfer.

6.4.2 What to encrypt

The loss of computer backup tapes is one type of event that triggers consumer notification. This has led to increasing data protection requirements.

What should you encrypt, and just as importantly, what should you not encrypt? There is an ever increasing focus on data security:

- California is generally considered the first state to implement a law requiring disclosure of security breaches in July 2003.
- Legislation has been enacted by 22 states that requires notification in cases of security breaches. See the following website for more information:

  `http://www.Privacyrights.org`
- Similar federal legislation has been proposed. See the following website for more information:

  `http://www.epic.org/privacy/bill_track.html`

Data protection requirements are driven by a variety of reasons. In addition to regulatory requirements that are driving the need for greater data security, integrity, retention, auditability, and privacy, reasons for increasing data protection are as follows:

- Severe business impacts that might be caused by loss or theft of data, including financial liability, reputation damage, legal risk, and compliance risk
- Increasing need to share data securely with IBM Business Partners and maintain backups at remote locations
- Need to reduce complexity and improve processes around enterprise encryption management
- Requirement to cost-effectively encrypt large quantities of tape data

There are additional drivers for encryption in financial services:

- A riskier environment

  Internet banking, for example, relies on open networks with multiple access points to conduct business in real time to drive down costs and improve response times to revenue generating opportunities.
- Growing regulatory burden, such as the following legislation:
  - Gramm-Leach-Bliley Act (GLBA) of 1999
  - California Law No. SB 1386
  - FCRA/FACTA amendments
  - Basel II

  Not all of the regulations specifically require the use of stored data encryption. However, many organizations are implementing encryption for their protected information in conjunction with other security layers to protect personally-identifiable information.
- Maturing industry standards, such as the Payment Card Industry (PCI) Data Security Standard (DSS)

6.4.3 Why use tape encryption

Tape encryption is used to hide and protect sensitive data. If tape data on cartridges leaves data centers, the data is no longer protected through RACF or similar access protection mechanisms. Tape encryption can help fulfill security regulations. Many governmental agencies require disclosure of security breaches. Industry organizations are also increasing their scrutiny of security procedures. Now, tape encryption uses an easy and economical way to protect data from unauthorized view.

Important and sensitive data can be protected in many ways. Data can be encrypted by means of special software programs, hardware adapters, facilities, or outside of the device where the data is stored. Encrypting data with software programs takes away processor power, and encrypting data with hardware requires additional investment in hardware for the computers.

The advantage of IBM Tape Encryption is that the data is encrypted after compression, and there are no additional software program costs. IBM Tape Encryption saves space on tape cartridges and saves additional hardware investments. In addition, outboard encryption in the tape drives might help you protect large volumes of tape data in a cost-effective way. Data on cartridges does not have to be degaussed or overwritten with patterns of x’FF’ at the end of life of the cartridge. This is valid for Write Once Read Many (WORM) cartridges and normal cartridges.

The new encryption key management capability is designed to manage keys across mainframes and open systems environments. There is only one component to be managed across multiple platforms.

Tape encryption can be managed by the applications or can be system-managed or library-managed.

Why encrypt data in the drive

The IBM Tape Drive Encryption solution encrypts the data within the drive using the 256-bit AES algorithm, rather than receiving previously encrypted data. There are several advantages to this system. By encrypting data in the drive, the drive can offer the most efficient data compression, because the drive first compresses the data, then encrypts it, providing more efficient data storage and media usage.

Encrypting in the drive also eliminates the need for any additional machines or appliances in the environment by offloading the encryption processing impact onto the drive. Because the drive can also process unencrypted workloads, the IT environment is further simplified, eliminating the need for separate drives to process data that does not need to be encrypted.

6.4.4 Summary

Encryption capability that is provided as a standard feature in the IBM TS1120 Tape Drive or the IBM LTO4 Tape Drive makes encrypting data stored on tape cartridges much easier. This situation is increasingly important as legislation continues to grow that requires notifying individuals when their personal information has potentially been compromised. The IBM developed tape drive-based encryption solutions described here, coupled with the new EKM component, enable key management and encryption in a wide variety of environments.

IBM provides tape drive encryption support in a wide range of operating systems environments:

- z/OS
- z/VM
- IBM z/VSE™
- z/TPF
- IBM System i5/OS®
- IBM AIX®
- Linux on System z
- Linux on other platforms
- HP-UX
- Sun Solaris
- Windows Server 2000 or Windows 2003 Server

For more information about tape encryption, see *IBM System Storage Tape Encryption Solutions*, SG24-7320, *IBM Virtualization Engine TS7500: Planning, Implementation, and Usage Guide*, SG24-7520 and the documentation available from the Encryption Facility for z/OS website, found at:

http://www.ibm.com/systems/z/os/zos/encryption_facility/

6.5 Overview of SSL and IP AT-TLS

SSL is a client-server cryptographic protocol based on the earlier SSL specifications developed by Netscape Corporation for securing communications that use TCP/IP sockets. The SSL protocol is executed at the application level. Therefore, an application has to be designed to support SSL to benefit from the SSL protection. When z/OS V1R7 Communications Server introduced Application Transparent-Transport Layer Security (AT-TLS), which implements TLS and its predecessor SSL at the Transport Control Protocol (TCP) layer of the TCP/IP stack, DB2 was able to provide secure (encrypted) communication for remote connections (inbound and outbound) by exploiting the functions of AT-TLS. In this book, the terms TLS and SSL are used interchangeably. The TLS/SSL service is available under z/OS and is called System SSL.

AT-TLS support is policy driven and is managed by the z/OS Communications Server Policy Agent (PAGENT), which implements policy-based networking for the z/OS environment. For information about policy-based networking, refer to *z/OS V1R10 Communications Server IP: Configuration Guide*, SC31-8775.

Because AT-TLS support is policy driven, it provides application-to-application security (such as TLS) using policies. As a result, DB2 continues to send and receive clear text data over the socket while an active AT-TLS policy enables TCP/IP to invoke the services of the System SSL to protect the data being transmitted over the network.

Figure 6-5 shows the basic flow of DB2 9 for z/OS and AT-TLS using SSL to protect the data exchanges over the network with a remote client system that also supports SSL.

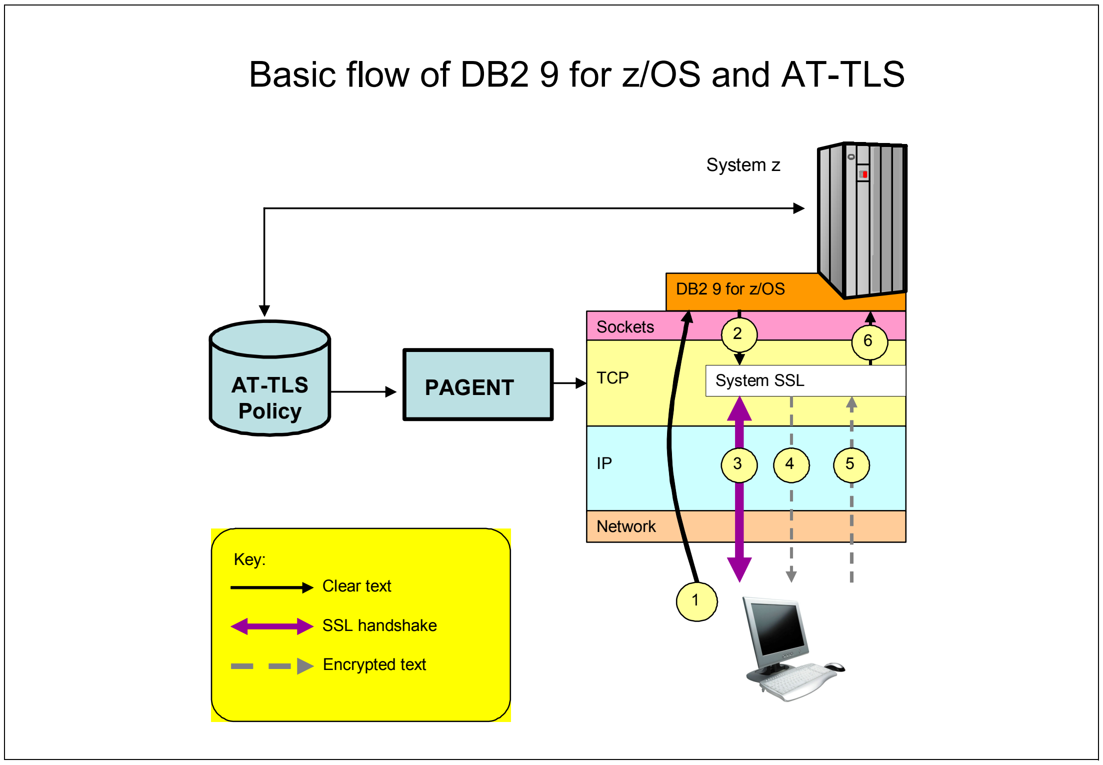


Figure 6-5 AT-TLS and DB2

Where:

1. The client connection to the DB2 server is established in the clear (there is no security, only a TCP handshake).
2. The DB2 server sends data in the clear and the TCP layer queues it.
3. The TCP layer invokes System SSL to perform an SSL handshake under the identity of the DB2 server, using policy information.
4. The TCP layer invokes System SSL to encrypt the queued data and sends it to the client.
5. The client sends encrypted data and the TCP layer invokes System SSL to decrypt the data.
6. The DB2 server receives data in the clear.

For details on how to setup and configure DB2 9 for z/OS with SSL using AT-TLS services, refer to *DB2 9 for z/OS: Configuring SSL for Secure Client-Server Communications,* REDP-4630.

7

User authentication

In today's world of open access, protecting your systems and data is more important than ever. Because the sources of many transactions are from untrusted networks such as the Internet, and sometimes unknown users, enterprises must pay increased attention to host and user authentication, data integrity and privacy in the network, and denial of service attacks.

A data server should have the following security functions:

- Authentication
- Authorization
- Encryption
- Auditing

Among these functions, authentication is one most easily being overlooked. However, because it provides the first gateway of security, stronger authentication approaches should be used to reduce the growing risk of widespread customer credential theft.

DB2 10 for z/OS integrates with the z/OS Security Server to provide more elaborate and encrypted authentication environments.

In this chapter, we talk about z/OS Security Server enhancements and how DB2 can support and use them to increase authentication level to keep your company‘s important information safe.

This chapter contains the following topics:

- Authentication and the data server security categories
- z/OS Security Server password phrase and DRDA encryption
- z/OS identity propagation and distributed DB2 workloads
- z/OS digital certificates and DB2 AT-TLS

These new functions are available after you have migrated to DB2 10 NFM.

7.1 Authentication and the data server security categories

Authentication is the validation process of identifying the requester. Authentication is responsible for the first layer of security control when a requester is accessing an System z server. The z/OS Security Server (long known as Resource Access Control Facility (RACF)) is in charge of authentication. Every requester who wants to use DB2 for z/OS should be authenticated by the z/OS Security Server.

Figure 7-1 illustrates the data server security categories and shows the position of authentication.

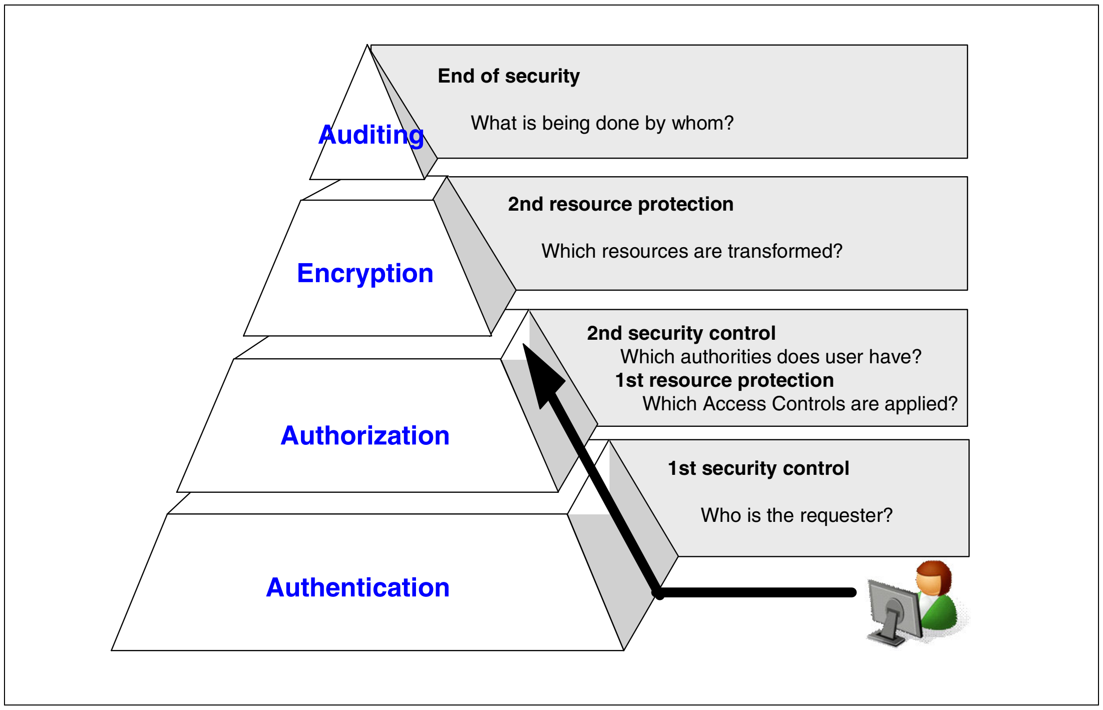


Figure 7-1 Data server security categories and position of authentication

DB2 for z/OS uses the z/OS Security Server for authentication and authorization to access any DB2 subsystem, which means that access for many resources is consistent, whether the resource is a file or communication or database access.

Another way of distinguishing the level of security is the software layer used for the access control. The tightest security would use a single security monitor. Using system controls and subsystems allows tighter security than having application programs provide the security.

DB2 10 for z/OS has added the SERVER_ENCRYPT option to the DSNZPARM TCPALVER[1] to enforce connection level security using strong authentication. If no authentication token is supplied and TCPALVER=SERVER_ENCRYPT, DB2 requires a user ID and a password. In addition, it requires that the security credentials be AES-encrypted or that the connection be accepted on a port that ensures AT-TLS policy protection, such as a DB2 Security Port (SECPORT). Kerberos tickets are accepted. RACF PassTickets, or non-encrypted security credentials, are accepted only when the connection is secured by the TCP/IP network.

[1] Defined in install panel DSNTIP5, DISTRIBUTED DATA FACILITY PANEL 2.

DB2 10 for z/OS integrates with the z/OS Security Server to provide more secure environments with:

- z/OS Security Server password phrase and DRDA encryption

  You can also use a long password phrase to log in to z/OS as a DB2 client, with your user ID and password, and even data, encrypted.

- z/OS identity propagation and distributed DB2 workloads

  Your system can have more delicate audit information using z/OS identity propagation.

- z/OS digital certificates and DB2 AT-TLS

  You can use z/OS client login by using a digital certificate. You access z/OS safely through the Secure Sockets Layer (SSL).

7.1.1 Support for a z/OS password phrase

The most common implementation is to challenge the requestor for a user ID and password. The user ID and password are static entities that generally do not change for an extended period of time, although some security policies can enforce a relatively frequent password change.

We all know that phishing, shoulder-surfing, and keylogging are simple but powerful methods to penetrate into the first security barrier of authentication. An intruder can easily acquire the authorization and privilege to access important information.

DB2 10 for z/OS supports RACF password phrases that were introduced in the z/OS Version 1 Release 10 Security Server feature. Password phrases have security advantages over traditional, 8-character passwords, because password phrases are more secure while being easier to remember.

A password phrase is a character string that consists of mixed-case letters, numbers, and special characters, including blanks. When the new-password-phrase exit (ICHPWX11) is present and allows it, a password phrase can be 9 to 100 characters in length. When ICHPWX11 is not present, the password phrase must be 14 to 100 characters in length.

We discuss DB2 10 DRDA-encrypted passwords or password phrases or DRDA-encrypted user IDs with encrypted passwords or password phrases in 7.2, “z/OS Security Server password phrase and DRDA encryption” on page 133.

Figure 7-2 shows the authentication process and what can be changed when we use password phrases.

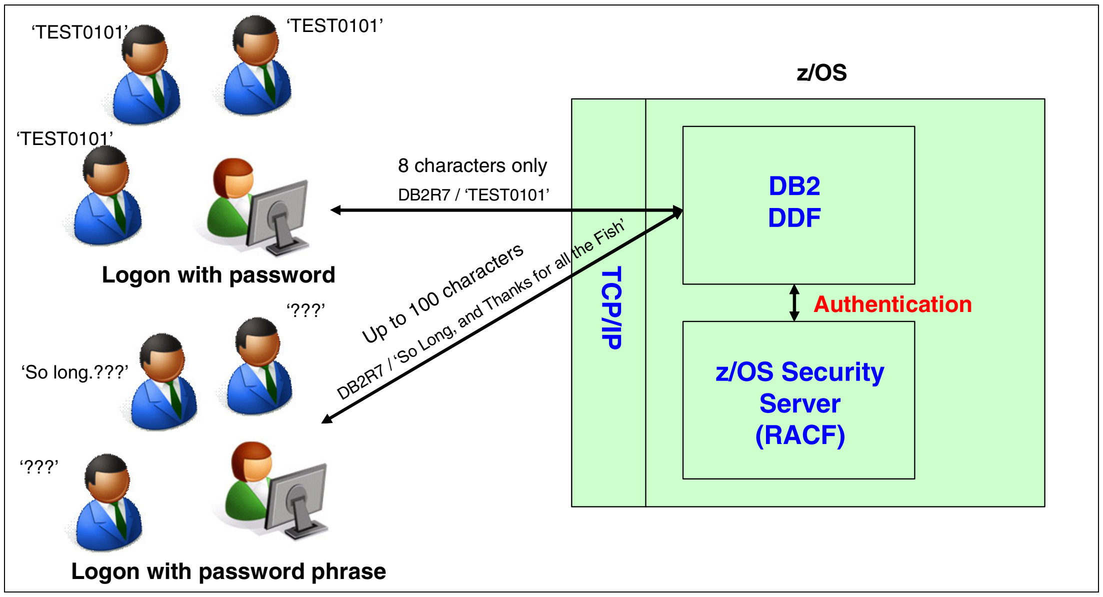


Figure 7-2 Comparison between using password and password phrase

7.1.2 Effectively audit distributed workloads

Transactions that run on a DB2 subsystem may originate outside of z/OS from the Internet and are initiated by users who authenticate their identities on web-based, or distributed, application servers.

When a distributed application server establishes a connection to a DB2 subsystem, the connection might be associated with the identity of the distributed application user, as defined in a user registry where the transaction originated, or it might be associated with a shared RACF user ID that was assigned by the z/OS subsystem.

To be effective, customers that audit user activities on DB2 subsystems need both the RACF user ID that is associated with the connection and the user identity that was presented when the user originally accessed the distributed application server.

Often instead, the connection is associated with the identity of the distributed application user, as defined in application server environment, or is associated with a shared RACF user ID. It means it is hard to distinguish between the RACF user ID and the user identity that was received when the user originally accessed the distributed application server.

Beginning with z/OS Version 1 Release 11, when you implement z/OS identity propagation, you can use the RACF RACMAP command to map the user's distributed identity to a RACF user ID. This mapping function is accomplished by using distributed identity filters. Also, RACF accepts information about the identities of distributed users from authorized applications that issue the RACROUTE REQUEST=VERIFY request.

This enhancement allows user identities to be recorded in the SMF audit records that are written by RACF during the execution of supported transactions and it provides more complete auditing for various z/OS subsystems, including DB2.

This distributed identity filters allow you to audit distributed workloads.

Figure 7-3 illustrates z/OS identity propagation.

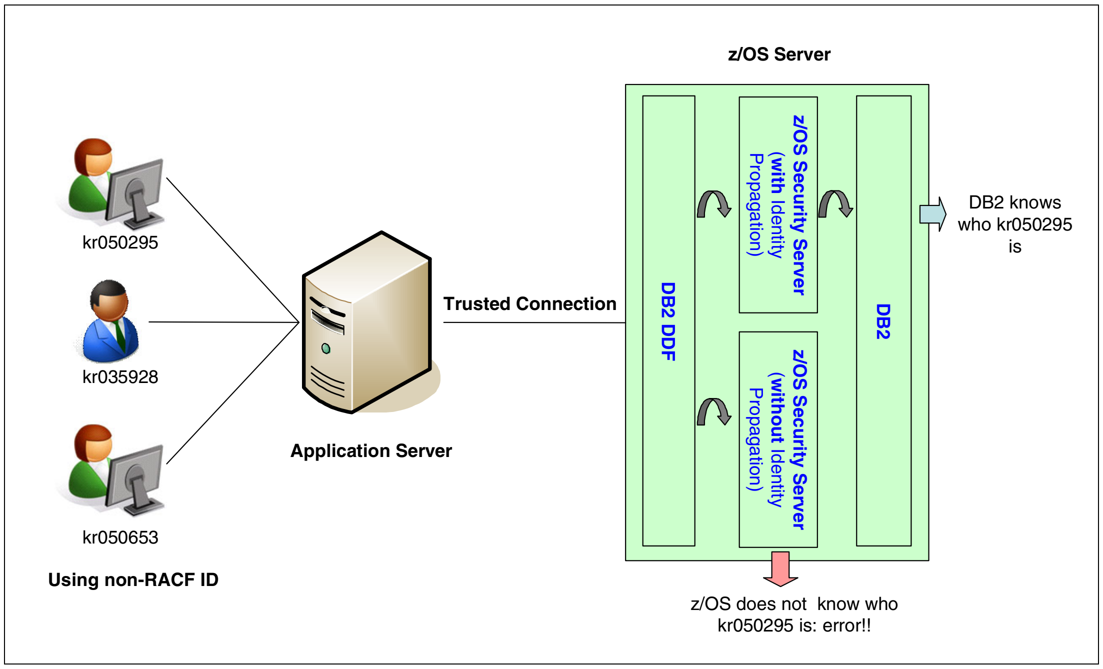

z/OS Server
kr050295
kr035928
kr050653
Using non-RACF ID
Application Server
Trusted Connection
DB2 DDF
z/OS Security Server (with Identity Propagation)
z/OS Security Server (without Identity Propagation)
DB2
DB2 knows who kr050295 is
z/OS does not know who kr050295 is: error!!

Figure 7-3 Comparison between with and without z/OS identity propagation

7.1.3 Support for z/OS client login using digital certificates

RACF users of traditional, remote client applications authenticated themselves to DB2 9 by providing their user IDs and passwords. With DB2 10, by using z/OS digital certificates, the Secure Sockets Layer (SSL) protocol supports server and client authentication during the handshake phase. This enhancement enables a server to validate the certificates of a client at the server, preventing the client from obtaining a secure connection without an installation-approved certificate. The authentication of the remote client's digital certificate is performed for DB2 by Application Transparent Transport Layer Security (AT-TLS) that is provided with the z/OS Communications Server TCP/IP stack.

Support for digital certificates also provides the benefit of RACF certificate name filtering. A certificate name filter enables you to associate many client certificates with one user ID based on the unique user information in the certificate, such as the user's affiliation. You can create one or more certificate name filters to map a large number of client certificates to a limited number of user IDs, which helps you reduce administrative costs.

Figure 7-4 shows how z/OS digital certificates work.

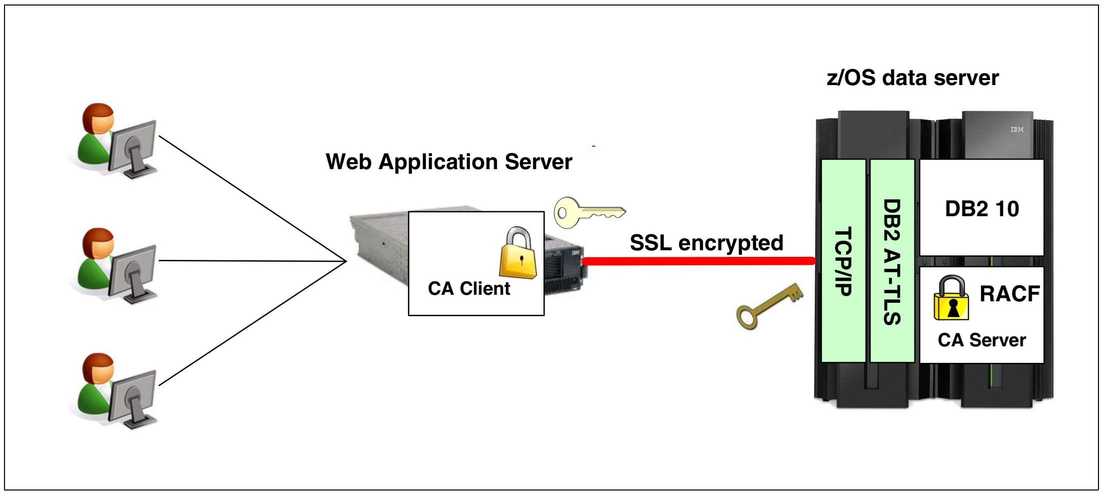


Figure 7-4 SSL connection with z/OS digital certificate

7.1.4 Using private protocol authorization

Private protocol was introduced in DB2 V2.2 to provide a connection between DB2 for z/OS subsystems, which is the first step towards distributed processing. It was also called system-directed access and provided support for the distributed unit of work (DUW). DRDA protocol, or application-directed access, became available a couple of years later with DB2 V2.3, providing support for the remote unit of work (RUOW).

Starting with DB2 6, DRDA had equivalent or better functions than private protocol. private protocol is officially deprecated in DB2 9.

The system parameter DBPROTCL[2] in the DSN6SYSP macro used to established your system default has been removed in DB2 9. You can still specify DBPROTOCOL(PRIVATE) rather than DBPROTOCOL(DRDA) at BIND/REBIND time, but DB2 issues a warning message if you perform any bind with this value. If DBPROTOCOL is not specified at BIND, you always get DRDA. The following APARs help with the elimination of private protocol:

- APAR PK92339 introduced a new DSNZPARM parameter on the DSN6FAC macro called PRIVATE_PROTOCOL. PRIVATE_PROTOCOL is online changeable by using the -SET SYSPARM command. The default value is YES, and NO disables private protocol.
- APAR PK64045 helped with the ALIAS resolution by providing the parameter DRDA_RESOLVE_ALIAS by using the DSNZPARM macro DSN6SPRM.
- APAR PM04968 supplied the routine DSNTIJPM that can be used to check DB2 9 and DB2 8 to see if they are DB2 10 for z/OS ready.

In DB2 10 for z/OS, private protocol is removed.

If DBPROTOCOL(PRIVATE) is specified, the Version 10 BIND or REBIND command parser issues error message DSNT225I and fails the BIND or REBIND request to prevent any outbound private protocol communications as a requester.

DSNT225I indicates that DBPROTOCOL(PRIVATE) is no longer a supported bind option.

[2] DATABASE PROTOCOL (DRDA or PRIVATE) on DDF installation panel DSNTIP5 with DB2 8.

Plans and packages that were previously bound using DBPROTOCOL(PRIVATE) should be converted to DRDA protocol before migration to Version 10. In Version 10, plans and packages that were bound with the DBPROTOCOL(PRIVATE) bind option and access remote locations cannot run. Applications that use packages or plans that were bound with DBPROTOCOL(PRIVATE) and access remote locations fail with SQLCODE -904. A rebind of those plans and packages must be explicitly performed before they can execute successfully. Job DSNTIJPM identifies the objects that must be converted to use DRDA protocol.

However, there are some lingering effects in terms of security when accessing a DB2 z/OS server from a DB2 z/OS requester.

Background of distributed security

There are fundamental differences about how authorization is performed based on the distributed protocol used:

- Private protocol
  - It supports only static SQL statements.
  - The plan owner must have the authorization to execute all SQL statements executed on the DB2 server.
  - The plan owner is authenticated on the DB2 requester (not on the DB2 server).
- DRDA protocol
  - It supports both static and dynamic SQL statements.
  - The primary auth ID or associated secondary auth IDs must have the authorization to execute the package and dynamic SQL statements on the DB2 server.
  - The primary auth ID is authenticated and the secondary auth IDs are associated on the DB2 server
- Up to DB2 9, private protocol and DRDA protocol can both be used by the same application.

Changes with DB2 10 APARs

DB2 10 APAR PM17665 provides a step forward for consistent and predictable authorization behavior for DB2 for z/OS DRDA servers regardless of the remote client type (DB2 for z/OS or any other).

The authorization behavior of a DB2 for z/OS DRDA client application was treated differently because it was still associated to a DB2 for z/OS plan, and thus DB2 for z/OS server processing applied the same authorization behavior as with the DB2 private protocol. In remote DB2 for z/OS DRDA, like private protocol, client application privileges were also inherited from the associated DB2 for z/OS plan owner ID. This privilege inheritance was honored by DB2 for z/OS servers relative to remote DB2 for z/OS client applications only.

DB2 for z/OS DRDA clients that access a DB2 for z/OS V8 or DB2 9 for z/OS server with private protocol disabled or DB2 10 for z/OS servers (where private protocol has been eliminated) might be affected by this change. After applying this APAR, access from remote DB2 for z/OS client applications through DRDA might fail with SQLCODE -551. This failure is likely to occur during the process of converting remote DB2 for z/OS applications from private protocol to DRDA protocol, but it may also occur if existing DRDA related application environments relied on the authorization behavior that has been removed. Authorization adjustments must be made to satisfy the condition described by the -551 SQLCODE.

In DB2 for z/OS V8 and V9, the DSN6FAC PRIVATE_PROTOCOL configuration parameter can be used to control the behavior. If the old authorization environment is required by remote DB2 for z/OS applications, the PRIVATE_PROTOCOL parameter may be set to YES; however, this setting also allows actual usage of private protocol.

APAR PM37300 provides a more flexible environment:

- It allows the usage of secondary group IDs.

  A primary authorization ID may be associated to secondary group IDs that are granted the necessary privileges, these secondary group IDs are recognized relative to non DB2 for z/OS remote client application environments, and the plan name associated with the remote DB2 for z/OS DRDA client applications will now be reported as DISTSERV.

- It introduces the DSNZPARM PRIVATE_PROTOCOL=AUTH parameter.

  - In DB2 for z/OS V8 and V9, the DSN6FAC PRIVATE_PROTOCOL parameter is extended to allow for a new AUTH keyword.

    Specifying AUTH indicates that private protocol is disabled yet still allows the old authorization behavior for the benefit of remote DB2 for z/OS DRDA client applications that may still rely on it.

  - In DB2 10 for z/OS, the PRIVATE_PROTOCOL parameter is being added to the DSN6FAC configuration macro.

    Specifying AUTH still allows the old authorization behavior for the benefit of remote DB2 for z/OS DRDA client applications that may still rely on it. Plan owner based package execution authorization is honored. Secondary group IDs are not used to determine package execution privileges for remote DB2 for z/OS applications. A specification of NO is the default.

    Specifying YES is not supported in DB2 10 for z/OS.

APAR PM38417 (currently open) provides further installation support for the new PRIVATE_PROTOCOL AUTH keyword and reverts to keeping the original DB2 plan name rather than changing it to DISTSERV.

Summary

After this maintenance, the following items apply for authorization checks performed when migrating from private protocol to the DRDA protocol:

- DB2 10 private protocol security semantics are no longer used as default for access from a DB2 z/OS requester.
- DB2 V8 and V9 use DRDA authorization checks when private protocol is disabled by setting system parameter DSN6FAC PRIVATE_PROTOCOL=NO.
- Before disabling private protocol, ensure all appropriate grants are performed by granting execute privilege to any user who plans to run a package or stored procedure package from a DB2 z/OS requester, just like other clients
- DB2 V8 and V9 can disable private protocol but maintain private protocol authorization checks by setting system parameter DSN6FAC PRIVATE_PROTOCOL=AUTH.
- DB2 10 does not support private protocol, but can allow private protocol authorization checks for use of DRDA protocol for DB2 z/OS requesters by setting system parameter DSN6FAC PRIVATE_PROTOCOL=AUTH.

7.2 z/OS Security Server password phrase and DRDA encryption

A z/OS password can be up to eight characters in length, but with a password phrase, you can use a memorable phrase up to 100 characters in length, which will make the password no longer vulnerable from the length point of view.

You can use upper case characters, lower case characters, and special characters, including spaces. Remember, it might be easily memorable to you because it is a phrase, but it is equally memorable for someone reading over your shoulder, so you need to combine upper, lower, and special characters appropriately.

You can see some sample test cases in Chapter 10, “Security”, in *DB2 10 for z/OS Technical Overview*, SG24-7892, as follows:

- Using SPUFI to connect to a remote location
- Using a JDBC type 4 application to connect to DB2 for z/OS
- Using an embedded static SQL connect statement from a COBOL program

Figure 7-5 shows a previous sample test case environment. It covers DB2 for z/OS client and a DB2 for z/OS server DRDA connection (1, SPUFI), a System z UNIX System Services JDBC Type 4 application’s request to DB2 for z/OS (2, JDBC Type 4), and a local COBOL (3. COBOL) program’s request to DB2 for z/OS.

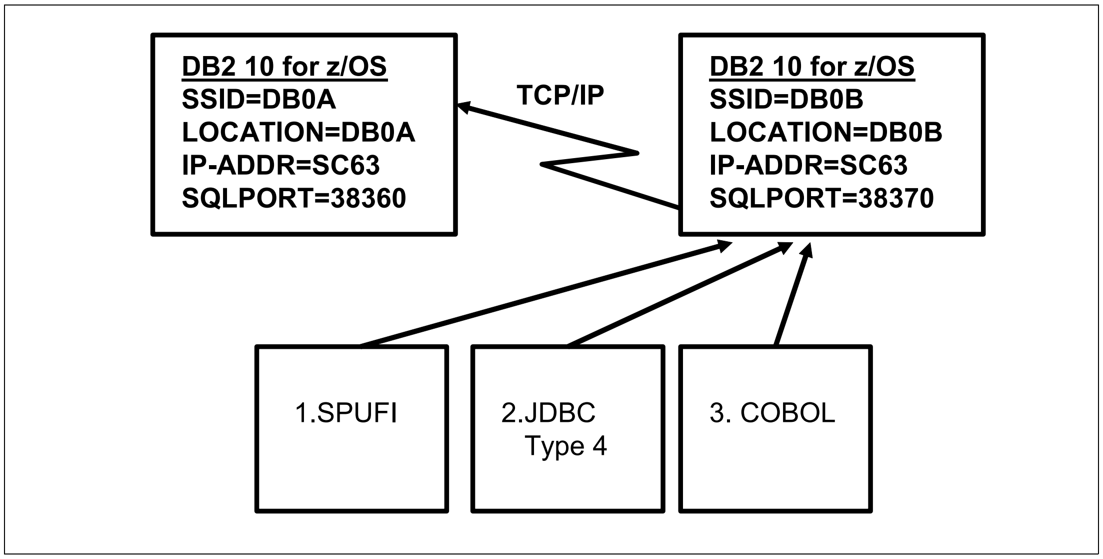


Figure 7-5 Sample DRDA test environment for password phrase scenario

For details about how to set up each scenario, read *DB2 10 for z/OS Technical Overview*, SG24-7892.

We now first examine a test case between a DB2 for z/OS client and another DB2 for z/OS server connection. As we know, we need to insert a password in the SYSIBM.USERNAMES, which is one of the CDB tables. The password column size has increased from VARCHAR(24) to VARCHAR(255) to support >100 characters in length.

In this case, note that, if you still do not use DSNLEUSR, you should consider using it right now to hide your password.

The DSNLEUSR stored procedure is a sample stored procedure that lets you store encrypted values in the translated authorization ID (NEWAUTHID) and password fields of the SYSIBM.USERNAMES table. This stored procedure also changed in DB2 10 to support a 100 character password.

Note: If you are still using unencrypted, transparent NEWAUTHID and PASSWORD in SYSIBM.USERNAMES, encrypt them using the DSNLEUSR stored procedure when you are communicating between DB2 systems.

Our scenario includes outside DRDA connections from a System z server using password phrase and a DB2 DRDA encrypted password or password phrase or include user ID.

This environment is basically DRDA communication between DB2 for z/OS in a System z environment and DB2 for LUW on Windows. We also test the connection between DB2 for z/OS and a WebSphere Application Server on Windows server.

Figure 7-6 illustrates the environment that we used to run the sample scenarios.

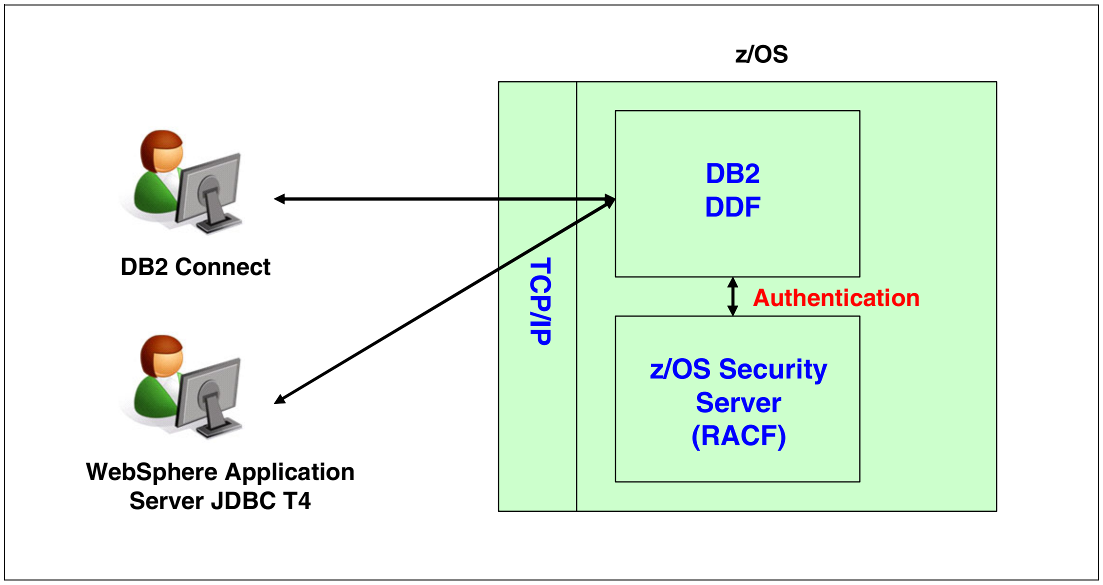

z/OS
DB2 DDF
TCP/IP
Authentication
z/OS Security Server (RACF)
DB2 Connect
WebSphere Application Server JDBC T4

Figure 7-6 DRDA password phrase scenarios test environment example

This environment includes the following items:

- System z environment
  - z/OS V1.12
  - DB2 10 for z/OS
- Windows environment
  - Windows XP
  - DB2 9.7.0.441
  - WebSphere Application Server - ND, 7.0.0.0

7.2.1 Creating a user ID with a password phrase

Here we created a simple user ID with a password phrase. We use the 'PHRASE' RACF sentence shown in Example 7-1 to create the user ID with a password phrase.

Example 7-1 RACF command to create a user ID with a password phrase

```
ADDUSER DB2R56 DFLTGRP(SYS1) OWNER(DB2R7) PHRASE('THIS IS INITIAL PWD')
```

Remember that the password phrase must be remembered easily, but other people can also do that, so you must create case sensitive and special characters in the password phrase. We then changed the initial password phrase to `'So Long, and Thanks for all the Fish'` using the DB2 command-line processor.

7.2.2 Connecting from a remote desktop to DB2 for z/OS using a password phrase

We catalog the DB2 for z/OS system for the first test and then we test the connection using the user ID and password phrase we created in 7.2.1, “Creating a user ID with a password phrase” on page 135.

Example 7-2 shows the sequence for the db2r56 user connecting to DB0A, the DB2 10 for z/OS subsystem for our test.

Example 7-2 Simple password phrase test to DB20A

```
db2 => connect to db0a user db2r56 using 'So Long, and Thanks for all the Fish'

   Database Connection Information

 Database server        = DB2 z/OS 10.1.5
 SQL authorization ID   = DB2R56
 Local database alias   = DB0A

db2 => select * from sysibm.sysdummy1
SQL0551N  "DB2R56" does not have the required authorization or privilege to
perform operation "EXECUTE PACKAGE" on object "NULLID.SQLC2H21".
SQLSTATE=42501
db2 => select * from sysibm.sysdummy1

IBMREQD
-------
Y

  1 record(s) selected.

db2 =>
db2 => select count(*) from sysibm.systables
SQL0551N  "DB2R56" does not have the required authorization or privilege to
perform operation "SELECT" on object "SYSIBM.SYSTABLES".  SQLSTATE=42501
```

We can see that password phrase can be used to connect to the DB2 10 for z/OS serve from the remote client.

However, when DB2R56 tries to read the SYSIBM.SYSDUMMY1 table, an error occurs because but there was no privilege to execute the SQLC2H21 package. After the package grants access to PUBLIC, DB2R56 has no problem reading SYSIBM.SYSDUMMY1, but DB2R56 now cannot read SYSIBM.SYSTABLES.

All these actions are audited by DB2 audit trace. Example 7-3 shows what was done by the DB2R56 remote user.

Example 7-3 DB2 Audit report for remote DRDA connection

```
LOCATION: DB0A                             OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)               PAGE: 1-1
      GROUP: N/P                                   AUDIT REPORT - DETAIL                        REQUESTED FROM: NOT SPECIFIED
     MEMBER: N/P                                                                                            TO: NOT SPECIFIED
  SUBSYSTEM: DB0A                             ORDER: PRIMAUTH-PLANNAME-REQLOC                      ACTUAL FROM: 04/29/11 16:35:49.12
DB2 VERSION: V10                                       SCOPE: MEMBER                                        TO: 04/29/11 16:57:17.00
PRIMAUTH CORRNAME CONNTYPE
ORIGAUTH CORRNMBR INSTANCE
PLANNAME CONNECT                TIMESTAMP   TYPE                                   DETAIL
-------- -------- ------------ ----------- -------- --------------------------------------------------------------------------------
DB2R56   028.DBAA 'BLANK'      16:35:49.12 AUTHCHG  TYPE:               END OF IDENTIFY                  STATUS:            SUCCESS
DB2R56     02     C7B218CED246                      PREVIOUS AUTHID:    DB2R56                           CURRENT SQLID:     DB2R56
'BLANK'  DB0A                                       SECONDARY AUTHID:   SYS1
REQLOC  :::9.30.28.115


SYSOPR   028.DBAA 'BLANK'      16:35:49.12 AUTHCHG  TYPE:               ENCRYPTED                        COMMS ADDR TYPE:   TCP/IP
SYSOPR     02     C7B218CED246                      IP ADDR:            000000000000000000000000091E1C73 PORT:              0553
'BLANK'  DB0A                                       DERIVED LOCAL UID:  db2r56                           CLIENT PRODUCT ID: SQL09073
REQLOC  :::9.30.28.115                              SECURITY MECHANISM: Encrypt UID PW.


DB2R56   db2bp.ex DRDA         16:36:01.59 AUTHFAIL AUTHID CHECKED: DB2R56               PRIVILEGE: EXECUTE
DB2R56   e        110429163550                      OBJECT TYPE   : PACKAGE              REASON:       0 RC: -    1
DISTSERV SERVER                                     SOURCE OBJECT : SQLC2H21             SOURCE OWNER:   NULLID
REQLOC  :::9.30.28.115                              TARGET OBJECT : N/A                  TARGET OWNER:   N/A
                                                    MLS    RID    : N/P                  SECLABEL:       N/P
                                                    TEXT:  N/P

ENDUSER :db2r56
WSNAME  :BRUNITEST
TRANSACT:db2bp.exe

DB2R7    DB2R7    TSO          16:37:48.42 AUTHCNTL GRANTOR: DB2R7      OWNER TYPE: PRIM/SECOND AUTHID
DB2R7    'BLANK'  C7B2190E8C4D                                          REASON: SYSADM                 SQLCODE:     0
ADB      TSO                                        OBJECT TYPE: PACKAGE
                                                    TEXT: GRANT EXECUTE ON PACKAGE "NULLID"."SQLC2H21" TO PUBLIC


DB2R56   db2bp.ex DRDA         16:38:22.00 AUTHFAIL AUTHID CHECKED: DB2R56               PRIVILEGE: SELECT
DB2R56   e        110429163550                      OBJECT TYPE   : TAB/VIEW             REASON:       0 RC: -    1
DISTSERV SERVER                                     SOURCE OBJECT : SYSTABLES            SOURCE OWNER:   SYSIBM
REQLOC  :::9.30.28.115                              TARGET OBJECT : N/A                  TARGET OWNER:   N/A
                                                    MLS    RID    : N/P                  SECLABEL:       N/P
                                                    TEXT: select count(*) from sysibm.systables

ENDUSER :db2r56
WSNAME  :BRUNITEST
TRANSACT:db2bp.exe
```

Description: DB2R56 was successfully identified and authenticated to DB0A with a password phrase and the DB2R56 user ID and password phrase were encrypted during this process.

Note that the DB2R56 user ID and password phrase were encrypted during identification and authentication. This is a DRDA-encrypted password or password phrase. In this case, it includes the user ID.

We now show an attempt by DB2R56 to try to connect to a DB2 9 for z/OS server using a password phrase in Example 7-4.

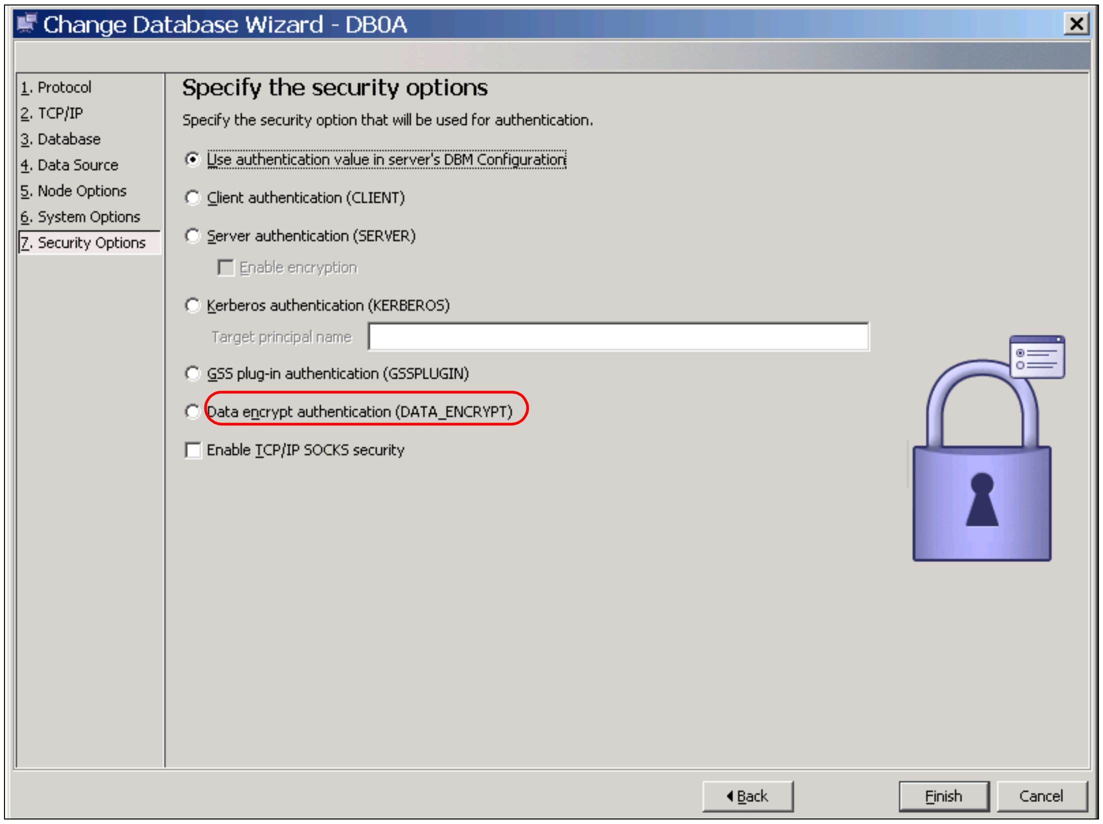
Example 7-4 Attempt to connect to a DB2 9 server using a password phrase

```
db2 => connect to db9a user db2r56 using 'So Long, and Thanks for all the Fish'
SQL30082N  Security processing failed with reason "24" ("USERNAME AND/OR
PASSWORD INVALID").  SQLSTATE=08001
```

DB2 9 does not support the password phrase, so if you have multiple DB2 subsystems and try to connect, they need to be DB2 10 servers.

Using a password phrase is a good decision for tightening security. A DRDA-encrypted password phrase can make the system safer. In addition, you can encrypt data as well as the user ID and password.

For data encryption enablement, you need to set up the security options shown in Figure 7-7.


Change Database Wizard - DB0A

1. Protocol
2. TCP/IP
3. Database
4. Data Source
5. Node Options
6. System Options
7. Security Options

Specify the security options

Specify the security option that will be used for authentication.

(●) Use authentication value in server's DBM Configuration
( ) Client authentication (CLIENT)
( ) Server authentication (SERVER)
[ ] Enable encryption
( ) Kerberos authentication (KERBEROS)
Target principal name
( ) GSS plug-in authentication (GSSPLUGIN)
( ) Data encrypt authentication (DATA_ENCRYPT)
[ ] Enable TCP/IP SOCKS security

Figure 7-7 DB2 for z/OS client setting for DRDA-encrypted user ID, password, and data

After checking **Data encrypt authentication (DATA_ENCRYPT)**, your user ID, password phrase, and data can be encrypted by DRDA, as shown in Example 7-5.

Example 7-5 DB2 Audit report for user ID, password , and data encrypted identification

```
DB2R56   028.DBAA 'BLANK'     22:08:24.93 AUTHCHG  TYPE:             END OF IDENTIFY                  STATUS:          SUCCESS
DB2R56     02     C7AE9D8DC1D9                     PREVIOUS AUTHID:  DB2R56                           CURRENT SQLID:   DB2R56
'BLANK'  DB0B                                      SECONDARY AUTHID: SYS1
REQLOC  :::9.30.28.127

SYSOPR   028.DBAA 'BLANK'     22:08:24.92 AUTHCHG  TYPE:             ENCRYPTED                        COMMS ADDR TYPE: TCP/IP
SYSOPR     02     C7AE9D8DC1D9                     IP ADDR:          0000000000000000000000000091E1C7F PORT:            0E46
```

```
'BLANK'  DBOB                               DERIVED LOCAL UID:  db2r56                      CLIENT PRODUCT ID: SQL09070
REQLOC  :::9.30.28.127                      SECURITY MECHANISM: Encrypt UID PW Data.
```

Now we have a DB2 connect remote connection to a DB2 for z/OS server using an encrypted user ID, password, and data.

7.2.3 Connecting JDBC type 4 using WebSphere Application Server to a DB2 for z/OS server using a password phrase

Using a JDBC type 4 connection with a password phrase uses exactly the same configuration setting method as not using the password phrase; we just add a user ID and a password phrase in the data source definition window.

First, we set up WebSphere Application Server through the administration console and test the connection to a DB2 for z/OS server using a password phrase.

For the JDBC type 4 connection to the DB2 for z/OS server, we need to download the JDBC license file from UNIX System Services, as shown in Example 7-6.

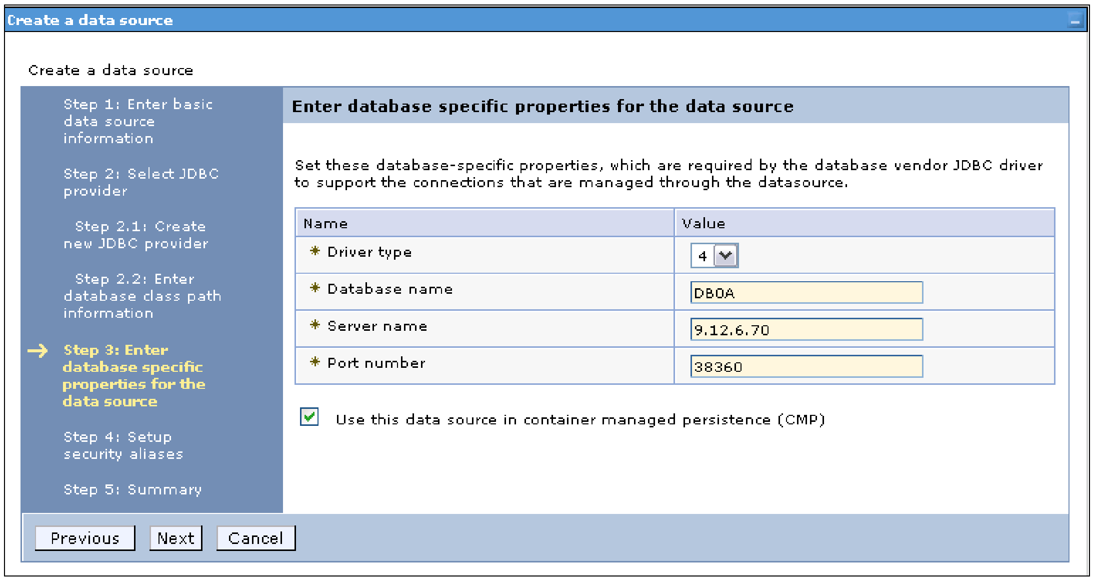
Example 7-6 JDBC license file

```
ftp> pwd
257 "/usr/lpp/db2/db0a/jdbc/classes" is the HFS working directory.
ftp> ls
200 Port request OK.
125 List started OK
IBM
db2jcc.jar
db2jcc4.jar
db2jcc_javax.jar
db2jcc_license_cisuz.jar
sqlj.zip
sqlj4.zip
250 List completed successfully.
ftp: 95 bytes received in 0.02Seconds 5.94Kbytes/sec.
ftp> get db2jcc_license_cisuz.jar
200 Port request OK.
125 Sending data set /usr/lpp/db2/db0a/jdbc/classes/db2jcc_license_cisuz.jar
250 Transfer completed successfully.
ftp: 2074 bytes received in 0.01Seconds 138.27Kbytes/sec.
ftp>
```

We copy the license file to the WebSphere Application Server server. In this case, the path for this file is `C:\Program Files\IBM\SQLLIB\java`.

Using the administration console (Integrated Solutions Console) of WebSphere Application Server, we can create the data source that contains the connection information to DB2 for z/OS by selecting **Resource** → **JDBC** → **Data sources** in the left pane of the window. In this case, the Driver type is 4 because this is a TCP/IP connection outside of the System z server. The rest is DB2 DDF information.

Figure 7-8 show the data source setup for DB2 for z/OS.


Create a data source
Create a data source
Step 1: Enter basic data source information
Step 2: Select JDBC provider
Step 2.1: Create new JDBC provider
Step 2.2: Enter database class path information
Step 3: Enter database specific properties for the data source
Step 4: Setup security aliases
Step 5: Summary
Previous
Next
Cancel

Enter database specific properties for the data source

Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through the datasource.

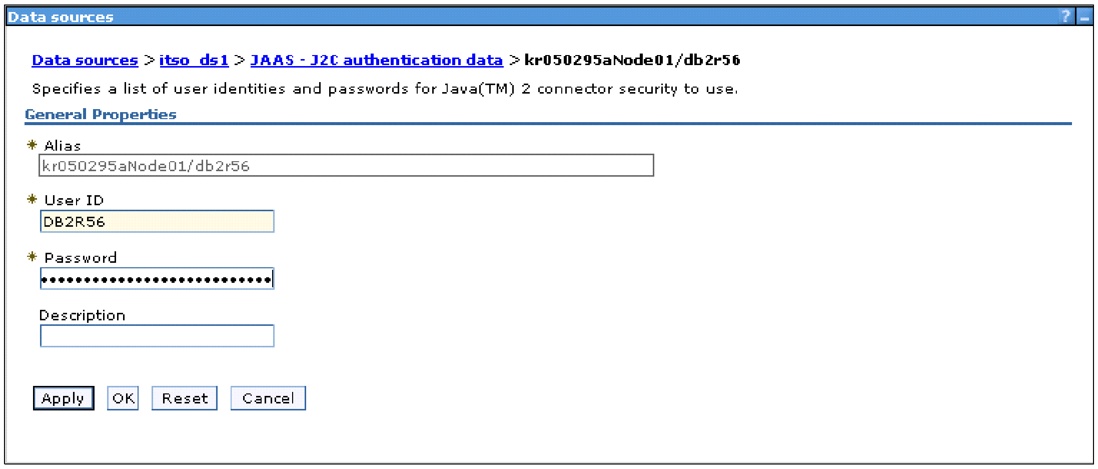
Name	Value
* Driver type	4
* Database name	DB0A
* Server name	9.12.6.70
* Port number	38360

✓ Use this data source in container managed persistence (CMP)

Figure 7-8 Create a data source

Each data source must have its own authentication information. We use a password phrase for this authentication information item. Use several data sources. Each data source has different authentication information, with different user IDs and password phrases, and also different minimized authorities. It is important to protect your data from an unexpected application security violation, such as exposing data source authentication information. Using different password phrases for multiple data sources prevents your data from unexpected exposure.

Figure 7-9 shows the password phrase used for data source authentication information of WebSphere Application Server.



Data sources > itso_ds1 > JAAS - J2C authentication data > kr050295aNode01/db2r56

Specifies a list of user identities and passwords for Java(TM) 2 connector security to use.

General Properties

* Alias	kr050295aNode01/db2r56
* User ID	DB2R56
* Password	••••••••••••••••••••••••••
Description	

Figure 7-9 Password phrase for data source authentication information

During logon to DB0A using DB2R56 using a password phrase, we can see the request of identification to DB0A through the DB2 audit report, as shown in Example 7-7.

Example 7-7 DB2 Audit report check for JDBC connection using a password phrase

```
PRIMAUTH CORRNAME CONNTYPE
ORIGAUTH CORRNMBR INSTANCE
PLANNAME CONNECT                TIMESTAMP   TYPE                                   DETAIL
-------- -------- ------------ ----------- -------- --------------------------------------------------------------------------------
DB2R56   028.DBAA 'BLANK'      21:42:16.30 AUTHCHG  TYPE:               END OF IDENTIFY                  STATUS:            SUCCESS
DB2R56     02     C7AFD9932DE6                      PREVIOUS AUTHID:    DB2R56                           CURRENT SQLID:     DB2R56
'BLANK'  DB0A                                       SECONDARY AUTHID:   SYS1
REQLOC  :::9.30.28.127
```

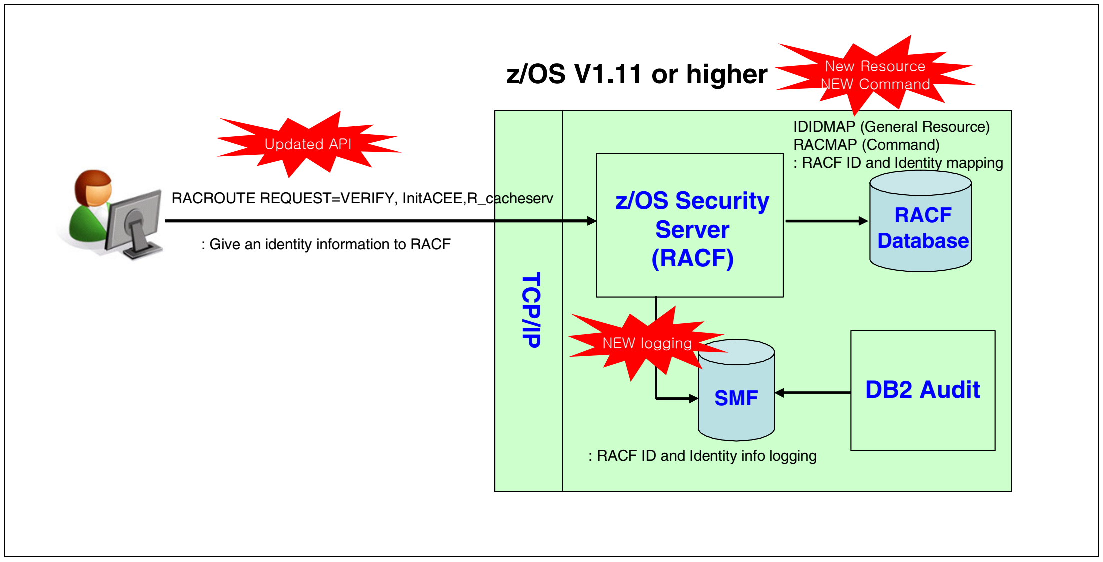
7.3 z/OS identity propagation and distributed DB2 workloads

Many distributed workloads are outside of a z/OS environment. z/OS has several mechanisms to deal with client identities similar to DB2 trusted context and roles, the SSL two-part authentication with digital certificates, and Kerberos principal and realms. z/OS V1.11 introduced identity propagation on z/OS Security Server. With DB2 9, enterprise identity mapping (EIM) enables the mapping of user identities across servers that are integrated but do not share user registries. With DB2 10, the EIM function is full integrated into the RACF database, so you no longer have to provide the LDAP server and EIM domain controller infrastructure that is required for the DB2 9 EIM implementation.

DB2 9 also introduced the ROLE authorization mechanism. You can define up to a 128 character identifier and assign it to remote or local connections using the DB2 trusted context and, optionally, activate the EIM function.

z/OS Security Server V1.11 includes this function, so a role with trusted context is easier to implement and you can have more detailed SMF audit records for general distributed workloads. z/OS identity propagation provides a consistent method of mapping distributed user identities into RACF user IDs by means of *distributed identity filters*. A distributed identity filter is a mapping association between a RACF user ID and one or more distributed user identities, because they are known to distributed application servers and are defined in distributed user registries. A distributed identity filter consists of one or more components of a distributed user's name and the name of the registry where the user is defined. To administer these distributed identity filters, use the RACF RACMAP command to associate (or map) a distributed user identity with a RACF user ID.

Figure 7-10 shows the key mechanism of how z/OS Security Server identifies the remote client and DB2 can audit it.



z/OS V1.11 or higher
New Resource
NEW Command
Updated API
IDIDMAP (General Resource)
RACMAP (Command)
: RACF ID and Identity mapping
RACROUTE REQUEST=VERIFY, InitACEE,R_cacheserv
: Give an identity information to RACF
z/OS Security Server (RACF)
RACF Database
TCP/IP
NEW logging
SMF
DB2 Audit
: RACF ID and Identity info logging

Figure 7-10   A new mechanism for recognizing identity information from a remote application

During DB2 remote connection authentication processing, when RACF accepts information about the identities of distributed users from DB2 (using RACROUTE REQUEST=VERIFY InitACEE, R_cacheserv), RACF attempts to derive the RACF user ID.

InitACEE is part of RACROUTE REQUEST RACF API and it adds a new IDID parameter that contains information about the distributed user's identity.

Let us look at the updated InitACEE parameter. Figure 7-11 might help you understand the mechanism.

Additional information: The Access Control Environment Element (ACEE) is the z/OS control block that represents a user's identity. It contains the user ID and other related information used in establishing the identity of the user and the user's credentials.

```
CALL IRRSIA00 (Work_Area,
               ALET, SAF_Return_Code,
               ALET, RACF_Return_Code,
               ALET, RACF_Reason_Code,
               Function_Code, Attributes, RACF_UserID,
               ACEE_Ptr, APPL_ID, Password,
               Logstring, Certificate, ENVR_In, ENVR_Out,
               Output_Area, X500 name, Variable_List,
               Security_Label, SERVAUTH_Name,
               Password_Phrase,
               IDID_Area )
```

Figure 7-11   Updated InitACEE parameter

There is a familiar parameter, the Password_Phrase, which is supported by DB2 10. Identity propagation can be supported by IDID_Area data, which is propagated from remote applications or servers.

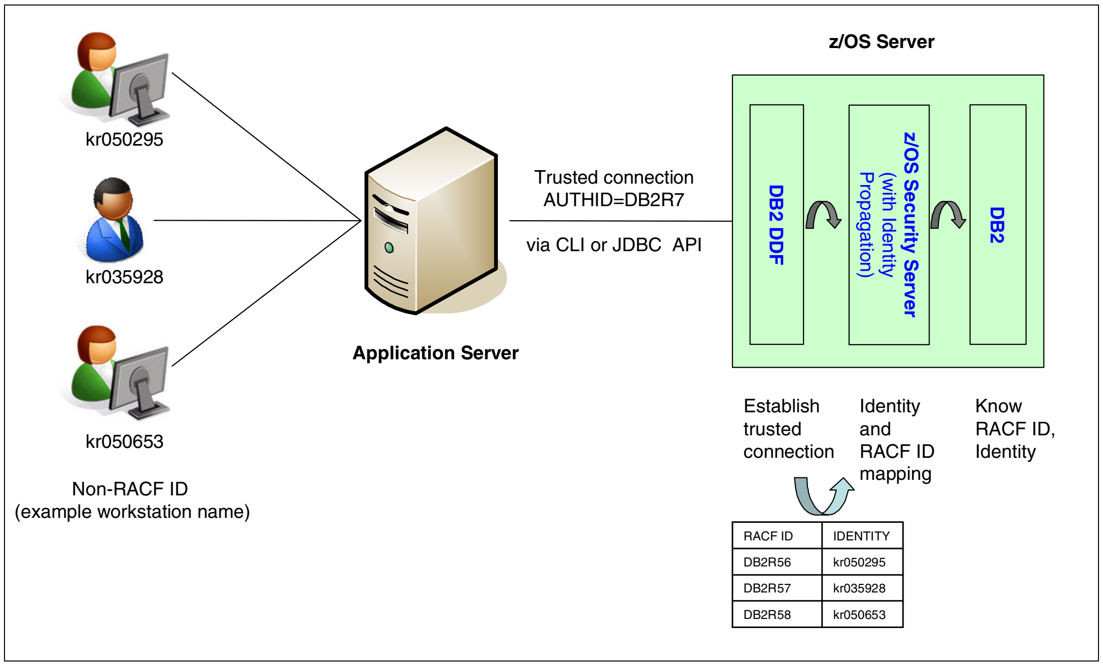
Authorized applications, such as DB2, can invoke the R_cacheserv callable service (IRRSCH00) to request the storage or retrieval of security-relevant information from a cache.

Using these enhanced parameters, RACF can create new data areas (IDID and ICRX), which are the base mechanisms that z/OS uses to propagate a remote user's identity.

7.3.1 Distributed identity propagation using DB2 10

DB2 10 supports z/OS identify propagation from the remote connection through application servers, but it should be done through a DB2 for z/OS supplied trusted connection. We introduce an example of a trusted connection through a CLI application and a JDBC application.

Figure 7-12 shows how to use z/OS identity propagation through a DB2 trusted connection.



RACF ID	IDENTITY
DB2R56	kr050295
DB2R57	kr035928
DB2R58	kr050653

Figure 7-12 DB2 trusted connection and RACF with identity propagation

Figure 7-13 shows how to use a DB2 trusted connection and RACF without using z/OS identity propagation.

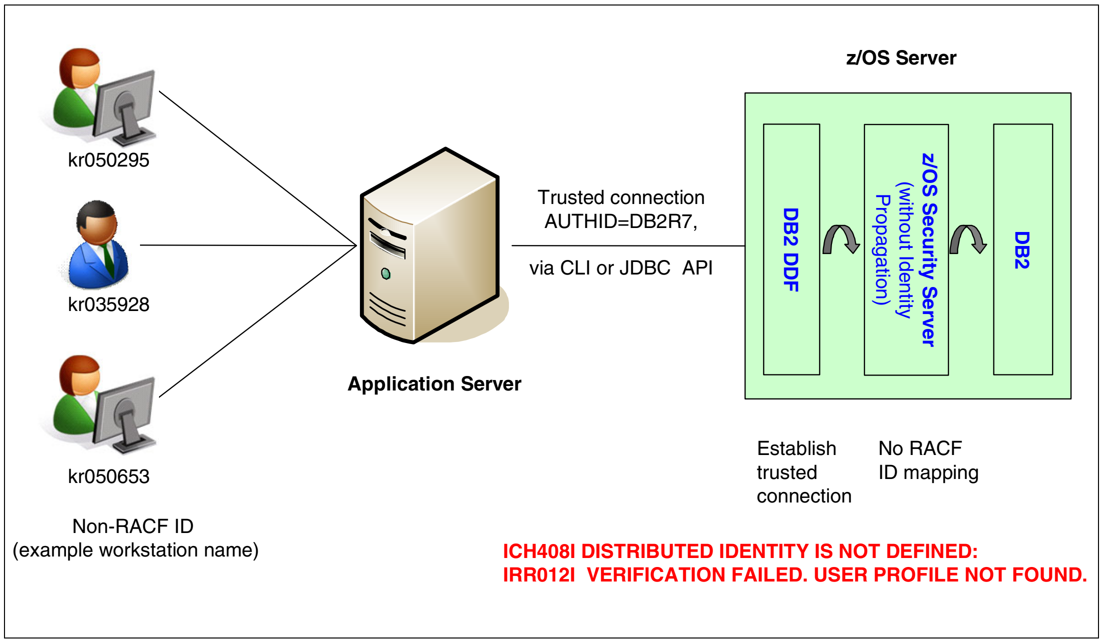

z/OS Server
kr050295
kr035928
kr050653
Non-RACF ID
(example workstation name)
Application Server
Trusted connection
AUTHID=DB2R7,
via CLI or JDBC API
DB2 DDF
z/OS Security Server
(without Identity Propagation)
DB2
Establish trusted connection
No RACF ID mapping
ICH408I DISTRIBUTED IDENTITY IS NOT DEFINED:
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.

Figure 7-13 DB2 trusted connection and RACF without identity propagation

For this implementation, three things should be done:

1. Perform RACF identity mapping.
2. Create a DB2 trusted context.
3. Perform application programming using DB2 provided a JDBC or CLI trusted connection API.

RACF identity mapping requires the following three steps:

- Activate a new RACF general resource class IDIDMAP.
- Perform identity mapping using the new RACMAP command.
- Refresh IDIDMAP.

The new RACF resource and command are shown in Figure 7-10 on page 141. To use them, complete the following steps:

1. Issue the RACMAP command with the MAP function:

   ```
   RACMAP ID(DB2R56) MAP
   USERDIDFILTER(NAME('kr050295'))
   REGISTRY(NAME('*'))
   WITHLABEL('DB2 10 default remote appl.')
   ```

   By specifying asterisk in the Registry Name, this filter will match an IDID that has any string in the Registry Name:

   ```
   REGISTRY(NAME('ldaps://us.ldapserv.com'))
   REGISTRY(NAME('ldap://12.34.56.78:389'))
   ```

2. Activate the IDIDMAP class and enable it for RACLIST processing by running the following command:

   ```
   SETROPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)
   ```

If the IDIDMAP class is already active and enabled for RACLIST processing, refresh the IDIDMAP class profiles by running the following command:

```
SETROPTS RACLIST(IDIDMAP) REFRESH
```

3. Review the new distributed identity filter by running the following command:

```
RACMAP ID(DB2R56) LISTMAP
```

If you want to delete mapping information, use the DELMAP syntax shown in Example 7-8. The ID and LABEL information must be provided.

Example 7-8 RACMAP command syntax

```
RACMAP
[ ID(userid) ]
   MAP
   USERDIDFILTER(NAME('distributed-identity-user-name' | '*' ))
   REGISTRY(NAME('distributed-identity-registry-name' | '*' ))
   [ WITHLABEL('label-name') ]
   | DELMAP
   [ (LABEL('label-name')) ]
   | LISTMAP
   [ (LABEL('label-name')) ]
```

As an alternative to using the MAP keyword, you could use the new QUERY function for the RACMAP command. This function was added in z/OS V1R13, with enhancements supporting z/OS identity propagation, and rolled back to releases R12 and R11 through APARs OA34258 (for RACF) and OA34259 (for SAF), as described at the following address:

ftp://ftp.software.ibm.com/s390/zos/racf/pdf/oa34258.pdf

The new keyword, QUERY, was added to help the RACF Security Administrators perform a reverse-look up where you know the UserDIDfiltername and Registry Name but do not know to what the RACF user ID is mapped.

The syntax of the QUERY keyword is similar to the MAP keyword:

```
RACMAP
   QUERY
   USERDIDFILTER(NAME('distributed-identity-user-name' | '*' ))
   REGISTRY(NAME('distributed-identity-registry-name' | '*' ))
```

The ID and WITHLABEL keywords are not used in this case. If they are specified, they are ignored and informational messages are issued.

After defining identity mapping in RACF, we need to create a trusted context in DB2 because this is the only way DB2 can use z/OS identity propagation. See Example 7-9.

Example 7-9 CREATE trusted context

```
CREATE TRUSTED CONTEXT CTXNAME1
       BASED UPON CONNECTION USING SYSTEM AUTHID DB2R7
       ATTRIBUTES (ADDRESS '9.30.28.127')
       ENABLE
       WITH USE FOR DB2R56,DB2R57,DB2R58;
```

You are now ready to use z/OS identity propagation with DB2m but you first need to use several APIs to implement a trusted connection to DB2 for z/OS.

The following APIs samples are about connecting to a DB2 trusted context or switching user IDs. After connecting, normal business application programming applies.

You need to establish an explicit trusted connection using the APIs in Table 7-1.

Table 7-1 Establish trusted connection APIs

Application type	**API**
CLI/ODBC	SQLConnect, SQLSetConnectAttr
XA CLI/ODBC	Xa_open
JAVA	getDB2TrustedPooledConnection, getDB2TrustedXAConnection

Example 7-10 and Example 7-11 on page 146 show examples of CLI and JDBC usage, which come from the Security section of “DB2 LUW V9R7 Database Fundamentals” at the following address:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

Example 7-10 CLI sample usage to establish a trusted connection

```
int main(int argc, char *argv[])
{
   SQLHANDLE henv;       /* environment handle */
   SQLHANDLE hdbc1;     /* connection handle */
   char origUserid[10] = "DB2R7";
   char password[10] = "test";
   char switchUserid[10] = "kr050295";
   char dbName[10] = "DBOA";

// Allocate the handles
SQLAllocHandle( SQL_HANDLE_ENV, &henv );
SQLAllocHandle( SQL_HANDLE_DBC, &hdbc1 );

// Set the trusted connection attribute
SQLSetConnectAttr( hdbc1, SQL_ATTR_USE_TRUSTED_CONTEXT, SQL_TRUE, SQL_IS_INTEGER
);

// Establish a trusted connection
SQLConnect( hdbc1, dbName, SQL_NTS, origUserid, SQL_NTS, password, SQL_NTS );

//Perform some work under user ID "DB2R7"
. . . . . . . . . . .

 // Commit the work
SQLEndTran(SQL_HANDLE_DBC, hdbc1, SQL_COMMIT);

// Switch the user ID on the trusted connection
SQLSetConnectAttr( hdbc1, SQL_ATTR_TRUSTED_CONTEXT_USERID, switchUserid,
SQL_IS_POINTER );

//Perform new work using user ID "kr050295" but system will know kr050295 is
mapping DB2R56
. . . . . . . . .

//Commit the work
SQLEndTranSQL_HANDLE_DBC, hdbc1, SQL_COMMIT);

// Disconnect from database
SQLDisconnect( hdbc1 );

   return 0;

}  /* end of main */
```

Example 7-11   JDBC sample usage to establish a trusted connection

```
/* Setting connection information*/
String databaseName = "DB0A" ;
String serverName = "9.12.6.70";
String driverType = "4";
String portNumber = "38360";
String user = "DB2R7";
String password = "test";

/* Establish Trusted Connection*/
DB2ConnectionPoolDataSource dataSource = new DB2ConnectionPoolDataSource();
      dataSource.setDatabaseName(databaseName);
      dataSource.setServerName(serverName);
      dataSource.setDriverType(Integer.valueOf(driverType));
      dataSource.setPortNumber(Integer.valueOf(portNumber));

      Properties properties = new Properties();

      Object[] objects = dataSource.getDB2TrustedPooledConnection(user, password, 
properties);

/* call is the cookie for the connection. Cast it as a byte array. */
byte[] cookie = ((byte[])(objects[1]);

/* Supply non-RACF user ID for the new connection and RACF id mapping registry*/
String newuser = "kr050295";
String userRegistry = "JimboRegistry";

/* Do not supply any security token data to be traced. */
byte[] userSecTkn = null;

/* Do not supply a previous user ID. */
String originalUser = null;

/* Define newuser's password but will not be used*/
String newpassword= null;

// Call getDB2Connection to get the connection object for the new user.
java.sql.Connection con = ((com.ibm.db2.jcc.DB2PooledConnection)pooledCon).
getDB2Connection(cookie,newuser,newpassword,userRegistry,userSecTkn,originalUser,p
roperties);
```

We map kr050295 to a specific user registry (we do not use the default asterisk registry).

We then run and check the distributed workload's identity propagation. Figure 7-14 shows the DB2 remote application identity mapping result.

```
1   LOCATION: DB0A                             OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)              PAGE: 1-1
       GROUP: N/P                                   AUDIT REPORT - DETAIL                     REQUESTED FROM: 05/24/11 22:42:00.00
      MEMBER: N/P                                                                                         TO: 05/24/11 23:59:00.00
   SUBSYSTEM: DB0A                                ORDER: PRIMAUTH-PLANNAME                       ACTUAL FROM: 05/24/11 22:42:30.08
 DB2 VERSION: V10                                       SCOPE: MEMBER                                     TO: 05/24/11 22:42:50.36
OPRIMAUTH CORRNAME CONNTYPE
 ORIGAUTH CORRNMBR INSTANCE
 PLANNAME CONNECT              TIMESTAMP   TYPE                                 DETAIL
 -------- -------- ------------ ----------- -------- --------------------------------------------------------------------------------
 DB2R56   db2jcc_a DRDA         22:42:30.19 AUTHCHG  TYPE:              END OF IDENTIFY                 STATUS:           SUCCESS
 kr050295 ppli     C7D1D965032A                      PREVIOUS AUTHID:    kr050295                        CURRENT SQLID:    DB2R56
 DISTSERV SERVER                                     SECONDARY AUTHID:   SYS1
 REQLOC  :::9.30.28.119
 ENDUSER :DB2R7
 WSNAME  :brenda
 TRANSACT:db2jcc_application

 DB2R56   db2jcc_a DRDA         22:42:30.19 AUTHCHG  TYPE:              REUSE TRUSTED CONTEXT           STATUS:           SUCCESS
 KR050295 ppli     C7D1D965032A                      OBJECT OWNER:       AUTHID                          SQLCODE:                0
 DISTSERV SERVER                                     SECURITY LABEL:
 REQLOC  :::9.30.28.119                              CONTEXT NAME:       CTXNAME1
 ENDUSER :DB2R7                                      CONTEXT ROLE:
 WSNAME  :brenda                                     USER ROLE:
 TRANSACT:db2jcc_application                         PREV. SYSAUTHID:    DB2R7
                                                     REUSE AUTHID:       DB2R56
                                                     SERVAUTH NAME:
```

Figure 7-14   DB2 remote application's identity propagation result

We can see that the trusted context was connected through DB2R7, the new user was changed to kr050295, and RACF identity propagation presents kr050295 as DB2R56 to DB2.

DB2 10 can now distinguish remote applications' identities by using the RACF identity propagation function.

7.3.2 Distributed identity propagation using CICS Transaction Gateway

We have several other ways to use z/OS identity propagation. For example, CICS Transaction Gateway for z/OS V8R0 can pass user security identity information (a distributed identity) from a J2EE client in WebSphere Application Server across the network to CICS Transaction Server for z/OS. The security identity of the user is preserved for use during CICS authorization and for subsequent accountability and trace purposes.

Identity propagation provides a way of authorizing requests by associating security information in WebSphere Application Server with security information in CICS Transaction Server for z/OS.

Identity propagation is supported on network topologies that use IP interconnectivity (IPIC) connections to CICS. If CICS Transaction Gateway and CICS Transaction Server are not on the same sysplex, SSL is required. A local mode topology is mandatory in this situation. If CICS Transaction Gateway and CICS Transaction Server are on the same sysplex, SSL is not required. Either a local mode or remote mode topology can be used in this situation. This IPIC protocol workload is eligible for an IBM eServer zSeries Application Assist Processor (zAAP) reroute.

Figure 7-15 shows a sample configuration to support z/OS identity propagation, including multiple CICS and DB2. For more details, see the CICS Transaction Gateway for z/OS Version 8 Release Information Center at the following address:

http://publib.boulder.ibm.com/infocenter/cicstgzo/v8r0/index.jsp?topic=/com.ibm.cics.tg.zos.doc/OpeningPagezos.htm

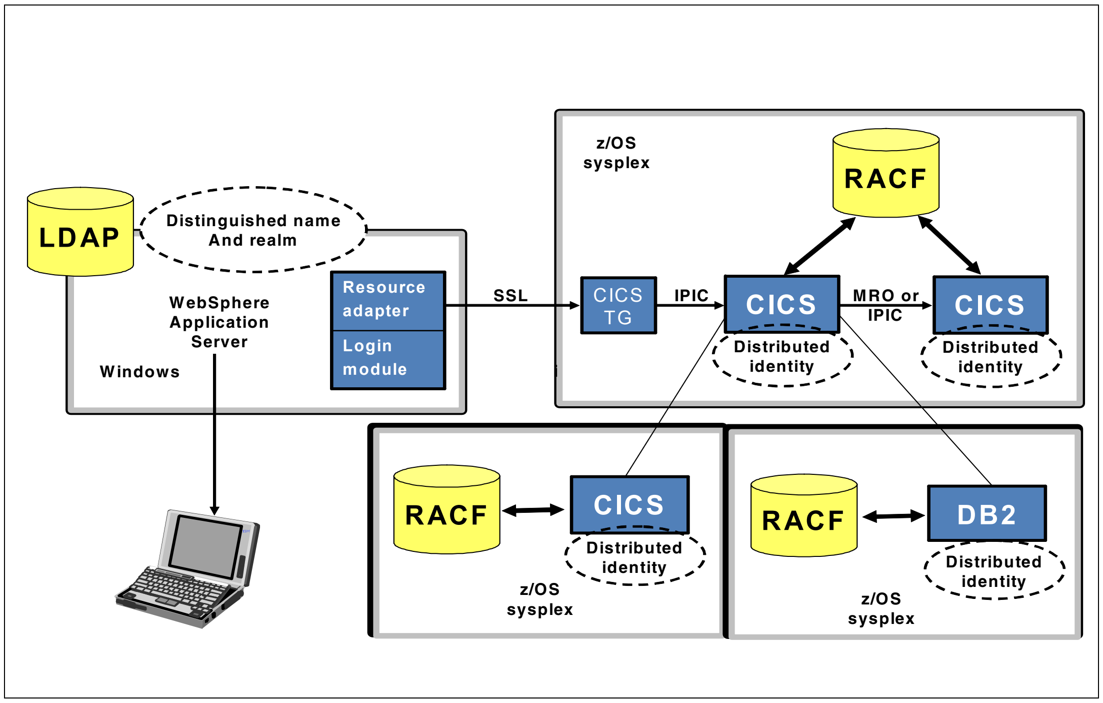


Figure 7-15 Example of identity propagation using CICS Transaction Gateway and DB2

In this case, you can map and distinguish between a non-RACF identity as defined in distributed LDAP registries and a RACF registered user ID so that System z can audit and manage from the user's point of view.

7.3.3 Distributed identity propagation with WebSphere Application Server V8

WebSphere Application Server V8 announced support for z/OS identity propagation, not only for DB2 for z/OS, but also for DB2 for LUW.

WebSphere Application Server V8 has enhanced security and auditability for applications requiring distributed and z/OS system access. If we use WebSphere Application Server V8 for a DB2 10 identity propagation test, we should not need to set an application to use the DB2 Trusted Connection API for CLI, JDBC, and so on.

Figure 7-16 shows how to set up WebSphere Application Server V8 for z/OS to identify z/OS.

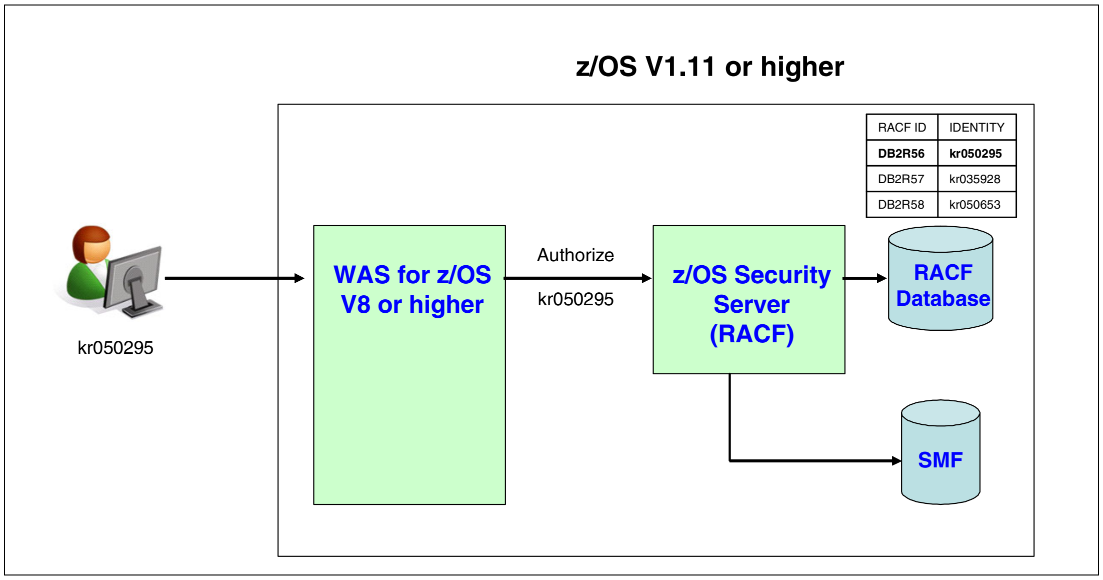


Figure 7-16 Example of identity propagation using WebSphere Application Server V8 for z/OS

Identifying distributed workloads is more and more important, not only for auditing purposes but also for management and operational purposes. Remote workload requirements keep increasing in the System z environment: z/OS identity propagation provides a mechanism to allow a user identity from an external security realm to be preserved, regardless of where the identity information was created, strengthening accountability across distributed environments. DB2, CICS CTG, CICS, and WebSphere Application Server support identity propagation to z/OS.

7.4 z/OS digital certificates and DB2 AT-TLS

Support for RACF client digital certificates is introduced in z/OS V1R10. It uses the SSL secure handshake to accomplish a user's identification and authentication to DB2. A DB2 client driver cannot present its certificate as identification and its proof-of-possession as authentication.

Previously, RACF users of traditional, remote client applications authenticated themselves to DB2 by providing their user IDs and passwords. Now, by using z/OS digital certificates, the Secure Sockets Layer (SSL) protocol supports server and client authentication during the handshake phase. This enhancement enables a server to validate the certificates of a client at the server, preventing the client from obtaining a secure connection without an installation-approved certificate. The authentication of the remote client's digital certificate is performed for DB2 by Application Transparent Transport Layer Security (AT-TLS), which is provided with the z/OS Communications Server TCP/IP stack.

Support for z/OS client login by using digital certificates also provides the benefit of RACF certificate name filtering. A certificate name filter enables you to associate many client certificates with one user ID based on the unique user information in the certificate, such as the user's affiliation. You can create one or more certificate name filters to map a large number of client certificates to a limited number of user IDs, which helps you reduce administrative costs.

The Secure Sockets Layer (SSL) protocol supports server and client authentication during the handshake phase.

The SSL provides server authentication as the minimum level of security. It uses the Server Authentication mechanism to secure communications between a server and its client and allows the client to validate the authenticity of the server.

DB2 10 support for SSL provides client authentication as an additional level of authentication and access control. It enables a server to validate the certificates of a client at the server and thus prevents the client from obtaining a secure connection.

For the implementation details, see 11.2, “Protecting data through DB2 SSL with digital certification” on page 241.

8

Audit policies

Prior to DB2 10, the collection of audit events was only controlled by using the DB2 START and STOP trace facility. In addition, there were restrictions surrounding the granularity of audit event data collected with audit trace.

In this chapter, we describe the improvements delivered in DB2 10, and describe how trace event collection can be controlled using DB2 10 audit policies, and, by using the new SECADM administrative authority, control of audit event collection can be removed from the traditional administration authorities such as SYSADM and DBADM.

8.1 Policy-based audit capability

DB2 provides a variety of authentication and access control mechanisms to establish rules and controls. However, to protect against and to discover and eliminate unknown or unacceptable behaviors, you need to monitor data access. DB2 10 assists you in this task by providing a powerful and flexible audit capability that is based on audit policies and categories, helping you to monitor application and individual user access, including administrative authorities. When used together with the audit filtering options introduced in DB2 9 for z/OS, policy-based auditing can provide granular audit reporting. For example, you can activate an audit policy to audit an authorization ID, a role, and DB2 client information.

The auditing capability is available in DB2 10 new-function mode (NFM) mode.

Prior to DB2 10, activity and data access was audited by using the audit trace classes and controlled by START and STOP trace commands. While providing some level of reporting capability, to balance the need for providing an audit trail while at the same time preserving application performance, some restrictions were placed on the DB2 audit trace that limited event granularity. This lack of granularity, for many regulatory compliance requirements, presented problematic characteristics and was viewed as lacking sufficient robustness. At the same time, for many customers, the management of the audit trace, the collection of audit event data, and the downstream reporting process were all functions controlled by the DBA. However, the audit reports being generated by this process were, in many cases, being used to audit the activities of these same DBAs. This situation represents a problematic scenario, which we refer to as *weak separation of roles and responsibilities*. Historically, the DB2 audit trace facility exhibited the following characteristics that were considered weak from a security standpoint:

- Audit traces could be started and stopped by DBA and system administrators.
- SQL events (Class 4 and 5) required a DBA to alter the target table to include the AUDIT attribute. When so altered, any packages or plans that reference the altered table were flagged as invalid, and suffered package invalidation.
- SQL events (Class 4 and 5) only collected the first SQL statement in each logical unit of work. The inability to guarantee collection of all events affecting an audited table is considered a security risk.

Much of the improvements to audit event collection provided by DB2 10 for z/OS revolve around these historical weaknesses.

8.1.1 Audit policies

An *audit policy* provides a set of criteria that determines the categories that are to be audited. Each category determines the events that are to be audited. You can define multiple audit policies based on your audit needs.

Each policy has a unique name assigned, which you use when you complete the following tasks:

- Create an audit policy by inserting a row into the SYSIBM.SYSAUDITPOLICIES table.
- Enable an audit policy by issuing a START trace command.
- Display the status of an activated audit policy by issuing a DISPLAY TRACE command.
- Disable an audit policy by issuing a STOP TRACE command.

Additional information: For information about how to collect DB2 traces in SMF or GTF, refer to *DB2 10 for z/OS Managing Performance,* SC19-2978.

Audit categories

There are several security events that must be audited to comply with laws and regulations and to monitor and enforce the audit policy. For example, auditing is usually performed for the following events:

- Changes in authorization IDs
- Unauthorized access attempts
- Altering, dropping, and creating database objects
- Granting and revoking authorities and privileges
- Reading and changing user data
- Running DB2 utilities and commands
- Activities performed under system administrative and database administrative authorities

DB2 10 groups these events into *audit categories*. Audit categories are supported in DB2 for LUW. The implementation in DB2 10 allows consistency through DB2 server platforms.

In DB2 10, you can use the audit categories shown in Table 8-1 to define audit policies.

Table 8-1 Audit categories

Audit category	Description
CHECKING	Denied access attempts due to inadequate DB2 authorization (IFCID140) and RACF authentication failures (IFCID 83).
CONTEXT	Utility start, objects or phase change, or utility end (IFCID 23, 24, 25).
DBADMIN	Generates IFCID 361 trace records when an administrative authority, DBMAINT, DBCTRL, DBADM, PACKADM, SQLADM, SYSTEM DBADM, DATAACCESS, ACCESSCTRL, or SECADM satisfies the privilege that is required to perform an operation. Database filtering can be performed through the DBNAME attribute for DBADM, DBCTRL, and DBMAINT authorities. All databases are audited if no database name is specified in the DBNAME Policy column. Collection ID filtering can be performed through the COLLID attribute for the PACKADM authority. All collection IDs are audited if no collection ID is specified in the COLLID column. If the DB2/RACF Access Control Authorization Exit is active, only the operations that are performed by the SECADM authority are audited.
EXECUTE	A trace record is generated for every unique table change or table read, including the SQL statement ID. The SQL statement ID is used in the IFCID 143 and IFCID 144 trace records to record any changes or reads by the SQL statement identified in the IFCID 145 trace records. One audit policy can be used to audit several tables of a schema by specifying the table names with the SQL LIKE predicate (for example TBL%). Only tables residing in universal table space (UTS), partitioned table spaces, and segmented table spaces can be audited. Table aliases cannot be audited. An audit policy can audit clone tables and tables that are implicitly created for LOBs or XML columns. Generates a trace record for all unique table access in a unit of work. The qualifying tables are identified in policy columns OBJECTSCHEMA, OBJECTNAME, and OBJECTTYPE. If the audit policy is started while an SQL query is running, access to the table will be audited for the next SQL query that accesses the table.
OBJMAINT	Table altered or dropped. Audit policy specifies the tables to be audited. One audit policy can be used to audit several tables of a schema by specifying the table names with the SQL LIKE predicate (for example TBL%). You can audit only tables defined in UTS, traditional partitioned table spaces, and segmented table spaces. An audit policy can also audit clone tables and tables that are implicitly created for XML columns. Audit object type are specified in the OBJECTTYPE column. A default OBJECTTYPE column value of blank means auditing of all supported object types.

Audit category	Description
SECMAINT	Grant and revoke privileges or administrative authorities (IFCID 141) and create and alter trusted contexts (IFCID 270) and roles. The SECMAINT category also includes IFCID 271 (create, alter, drop row permissions and column masks).
SYSADMIN	Generates IFCID 361 trace records when an administrative authority, installation SYSADM, installation SYSOPR SYSCTRL, or SYSADM satisfies the privilege required to perform an operation. If the DB2/RACF Access Control Authorization Exit is active, only the operations that are performed by the installation SYSADM and installation SYSOPR authorities are audited.
VALIDATE	New or changed authorization IDs (IFCID 55, 83, 87, 169, 269, and 312), and establishment of new or user switching within a trusted connections

Tip: For details about DB2 10 audit categories, refer to *DB2 10 for z/OS Administration Guide,* SC19-2968.

Creating an audit policy in SYSIBM.SYSAUDITPOLICIES

You create an audit policy by inserting a row into the SYSIBM.SYSAUDITPOLICIES catalog table. Each row in SYSIBM.SYSAUDITPOLICIES represents an audit policy. An audit policy can be identified uniquely by its policy name.

The policy name is stored in the AUDITPOLICYNAME table column. This column is also the unique index key. SYSIBM.SYSAUDITPOLICIES provides one column for each audit category.

An audit category is selected for auditing by populating its corresponding category policy column. The columns of the SYSIBM.SYSAUDITPOLICIES table and its valid category column values are listed in Table 8-2.

Table 8-2 The SYSIBM.SYSAUDITPOLICIES catalog table

Column name	Data type	Description
ALTERDTS	TIMESTAMP NOT NULL WITH DEFAULT	Alter time stamp.
AUDITPOLICYNAME	VARCHAR(128) NOT NULL	Policy name.
CHECKING	CHAR(1) NOT NULL WITH DEFAULT	► A: Audit all failures. ► Blank: None IFCID 140.
COLLID	VARCHAR(128) NOT NULL WITH DEFAULT	► Name of package collid. ► If no collid is specified, all collids are audited. ► Applies only to authority PACKADM in category DBADMIN.
CONTEXT	CHAR(1) NOT NULL WITH DEFAULT	► A: Audit all utilities. ► Blank: None. IFICD 23, 24, 25.
CREATEDTS	TIMESTAMP NOT NULL WITH DEFAULT	Insert time stamp.

Column name	Data type	Description
DB2START	CHAR(1) NOT NULL WITH DEFAULT	Automatic policy start at DB2 startup, except for DB2 restart light. Up to 8 policies can be started at DB2 startup. ► Y: Automatic start. ► N: No automatic start. ► S: Automatic start as secure.
DBADMIN	VARCHAR(128) NOT NULL WITH DEFAULT	Database administrative tasks. ► *: Audit all. ► B: SYSTEM DBADM. ► C: DBCTRL. ► D: DBADM. ► E: SECADM. ► G: ACCESSCTRL. ► K: SQLADM. ► M: DBMAINT. ► P: PACKADM. ► T: DATAACCESS. ► Blank: Audit none. Can be a concatenated string of multiple parameters, IFCID 361.
DBNAME	VARCHAR(24) NOT NULL WITH DEFAULT	► Database name. ► If no database is specified, all databases are audited. ► Applies only to authorities DBADM, DBCTRL, DBMAINT authorities in column DBADMIN.
EXECUTE	CHAR(1) NOT NULL WITH DEFAULT	Tables qualifying through OBJECTSCHEMA, OBJECTNAME, and OBJECTTYPE will be audited. ► A: Audit any SQL activity of that application process. ► C: Audit SQL insert, update, or delete of that application process. ► Blank: None. Traces bind time information about SQL statements qualified by OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE. IFCID 143, 144, 145.
IBMREQD	CHAR(1) NOT NULL	Dependency indicator. A value of Y indicates that the row came from the basic machine-readable material (MRM) tape.
OBJECTNAME	VARCHAR(128) NOT NULL WITH DEFAULT	Object name. Applies only to categories OBJMAINT, EXECUTE. Can be specified in SQL LIKE notation.
OBJECTSCHEMA	VARCHAR(128) NOT NULL WITH DEFAULT	Schema name. Applies only to categories OBJMAINT and EXECUTE.
OBJECTTYPE	CHAR(1) NOT NULL WITH DEFAULT	► C: Clone Table ► P: Implicit table for XML column ► T: Table ► Blank: All Applies only to OBJMAINT and EXECUTE categories.

Column name	Data type	Description
OBJMAINT	CHAR(1) NOT NULL WITH DEFAULT	► A: Audit alter or drop activities when a table qualifies through OBJECTSCHEMA, OBJECTNAME, and OBJECTTYPE. ► Blank: None. IFCID 142.
SECMAINT	CHAR(1) NOT NULL WITH DEFAULT	► A: Audit all. ► Blank: None. IFCID 141 grant, revoke. IFCID 270 trusted context created or altered.
SYSADMIN	VARCHAR(128) NOT NULL WITH DEFAULT	System administrative tasks: ► *: Audit all. ► I: Install SYSADM. ► L: SYSCTRL. ► O: SYSOPR. ► R: Install SYSOPR. ► S: SYSADM. ► Blank: Audit none. Can be a concatenated string of multiple parameters, IFCID 361.
VALIDATE	CHAR(1) NOT NULL WITH DEFAULT	► A: Audit all ► Blank: None Assignment or change of authorization ID: IFCID 55, 83, 87, 169, 312. Trusted context establishment or user switch: IFCID 269.

Catalog tables: For details about all catalog tables and their columns, refer to the appendix “Additional information for DB2 SQL, in *DB2 10 for z/OS SQL Reference,* SC19-2983.

Added and changed instrumentation facility component identifiers

Table 8-3 lists the instrumentation facility component identifiers (IFCIDs) that are added or changed to support DB2 audit policies and the administrative authorities.

Table 8-3 New and changed IFCIDs

IFCID	Description
106	System parameters in effect. Changed to account for the following DSNZPARMs: ► SECADM1 ► SECADM2 ► SECADM1_TYPE ► SECADM2_TYPE ► SEPARATE_SECURITY ► REVOKE_DEP_PRIVILEGES
140	Authorization failures. Changed to include constants for new authorities.
141	Explicit grants and revokes. Changed to include constants for new authorities.
143	Changed to include a unique statement identifier.

IFCID	Description
144	Changed to include a unique statement identifier.
145	Changed to trace the entire SQL statement and to include the unique statement identifier.
361	Added to support auditing of administrative authorities.
362	Added for auditing the START AUDIT TRACE command. This trace is started automatically when a START TRACE command with the AUDTPLCY keyword is issued. IFCID 362 documents whether an audit policy was started successfully

Additional information: For a complete list of DB2 10 IFCIDs, refer to *DB2 10 for z/OS What's New*, GC19-2985.

APAR PM28296 introduced some enhancements to audit policies capabilities:

- While audit policies provide the ability to monitor application and individual user access, including administrative authorities, security administrators with SECADM authority can create those security policies and collect audit traces. Before this change, any user with TRACE privilege would have been able to stop the audit traces using the STOP TRACE command. This can be a security vulnerability. The APAR introduced a new value S to the DB2START column of the SYSIBM.SYSAUDITPOLICIES table, which enables audit trace during the DB2 startup and can only be stopped by a user with SECADM authority or by a DB2 shutdown. If the audit policies have already been started, then after updating the DB2START column, the trace needs to be stopped and restarted to get the secure behavior.
- The SECMAINT category had been changed to include IFCID 271, which audits row permission and column mask DDLs.

Refer to *DB2 10 for z/OS Technical Overview*, SG24-7892 and *DB2 10 for z/OS Administration Guide*, SC19-2968 for detail about the audit policies tasks and audit categories.

Audit trace and audit categories

Table 8-4 shows the mapping between audit trace classes, the instrumentation facility component identifiers (IFCIDs), and the new audit categories. The previous audit classes and the new audit policies show different auditing functionality and are grouped differently.

Table 8-4 Mapping of audit classes to audit policies

Audit class	IFCID	Audit category for audit policies	Notes on audit policies
1	140	CHECKING	
2	141	SECMAINT	Also includes IFCIDs 270 and 271.
3	142	OBJMAINT	
4	143	EXECUTE	Records all changes and reads. Can either audit all SQL statements or INSERT, UPDATE, or DELETE only.
5	144		
6	145		

Audit class	IFCID	Audit category for audit policies	Notes on audit policies
7	55, 83, 87,169, and 319	VALIDATE	Also includes IFCIDs 269 and 312
8	23, 24, 25, 219, and 220	CONTEXT	Excludes IFCID 219s and 220
9	146		
11	361	DBADMIN SYSADMIN	Can audit specific authorities.

Policy name considerations

The SYSIBM.SYSAUDITPOLICIES catalog table has a unique key constraint defined on the AUDITPOLICYNAME column. If you try to insert a policy row using an AUDITPOLICYNAME value that already exists in the table, you receive SQLCODE -803 (duplicate key) as you would for any other user table. Thus, plan your naming standards for audit policy names carefully.

Creating audit policy samples

We created three audit policies for use and reference throughout this chapter. Example 8-1 and Example 8-2 show the SQL insert statements that we used for policy creation.

Auditing all access to the DB2R5.EMP and DB2R5.DEPT tables

Example 8-1 inserts a policy row into SYSIBM.SYSAUDITPOLICIES. The table row provides the following attribute settings:

AUDITPOLICYNAME	AUDEMP and AUDDEPT
OBJSCHEMA	DB2R5
OBJNAME	EMP and DEPT
OBJTYPE	T to indicate table type
EXECUTE	A to audit all access

Example 8-1 Audit all SQL for multiple tables

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
VALUES('AUDEMP','DB2R5','EMP','T','A');
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
VALUES('AUDDEPT','DB2R5','DEPT','T','A');
```

Audit policy to start automatically during DB2 start

Example 8-2 updates the AUDSYSADMIN audit policy in the SYSIBM.SYSAUDITPOLICIES table to mark the policy for automatic start during DB2 startup.

Example 8-2 Mark audit policy for automatic start during DB2 start

```
UPDATE SYSIBM.SYSAUDITPOLICIES
  SET DB2START = 'Y'
  WHERE AUDITPOLICYNAME = 'AUDSYSADMIN';
```

Important: Only the install SYSADM and the SECADM authority have the privilege to insert, update, or delete rows in SYSIBM.SYSAUDITPOLICIES and only SECADM can grant these privileges to others.

If DSNZPARM SEPARATE_SECURITY is set to NO, the SYSADM authority implicitly holds the SECADM system privilege and can, therefore, perform any work that requires SECADM authority.

For simplicity, we refer only to the SECADM authority when we discuss the privilege of being able to modify SYSIBM.SYSAUDITPOLICIES as SECADM, install SYSADM, or SYSADM with DSNZPARM SEPARATE_SECURITY set to NO.

8.1.2 Starting, stopping, and managing DB2 audit policies

Policy-based auditing is operated using start, stop, and display audit trace commands. In DB2 10, these commands are enhanced to support an AUDTPLCY parameter (see Figure 8-1). Using the policy name to manage audit traces provides additional simplification, because DB2 chooses the IFCID information that is appropriate for the selected audit policy.

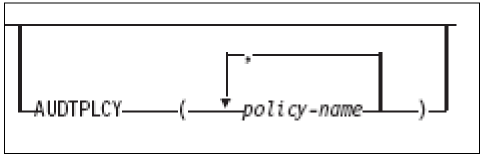


Figure 8-1 Start audit trace parameter AUDTPLCY

The AUDTPLCY parameter can take multiple policy names. Provide as many audit policies as possible in the AUDPLCY keyword, because DB2 starts only one trace for all policies that are specified, which helps to avoid the system limit of up to 32 started traces. However, you cannot stop an individual audit policy if that policy was started with other policies within the AUDTPLCY keyword. The AUDTPLCY keyword is mutually exclusive with the IFCID or CLASS keywords.

In the scenario shown in Figure 8-2, we start the two audit policies that we defined in Example 8-1 on page 158 within one single START TRACE command. We then tried to stop just the AUDEMP audit policy, and the command failed with DB2 message DSNW137I.

```
-START TRACE(AUDIT) AUDTPLCY(AUDEMP,AUDDEPT)
....
-STOP  TRACE(AUDIT) AUDTPLCY(AUDEMP)
DSNW137I  -DB0B  SPECIFIED TRACE NOT ACTIVE
DSN9022I  -DB0B DSNWVCM1 '-STOP TRACE' NORMAL COMPLETION
-STOP  TRACE(AUDIT) AUDTPLCY(AUDEMP,AUDDEPT)
DSNW131I  -DB0B STOP TRACE SUCCESSFUL FOR TRACE NUMBER(S) 03
DSN9022I  -DB0B DSNWVCM1 '-STOP TRACE' NORMAL COMPLETION
-START TRACE(AUDIT) AUDTPLCY(AUDDEPT) DEST(GTF)
DSNW130I  -DB0B AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSNW192I  -DB0B AUDIT POLICY SUMMARY
AUDIT POLICY AUDDEPT STARTED
END AUDIT POLICY SUMMARY
DSN9022I  -DB0B DSNWVCM1 '-START TRACE' NORMAL COMPLETION
```

Figure 8-2 Use of AUDTPLCY in start and stop trace commands

Starting, displaying, and stopping audit trace samples

We ran the following commands to start, display, and stop the audit policies:

- Start the AUDSYSADMIN audit policy successfully for the DB0BSYSADM role after the policy is defined (Figure 8-3). The audit trace records are written to destination GTF, which requires the GTF trace for DB2 to be started.

```
-START   TRACE(AUDIT) AUDTPLCY(AUDSYSADMIN) DEST(GTF) ROLE(DB0BSYSADM)
DSNW130I  -DB0B AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 04
DSNW192I  -DB0B AUDIT POLICY SUMMARY
AUDIT POLICY AUDSYSADMIN STARTED
END AUDIT POLICY SUMMARY
DSN9022I  -DB0B DSNWVCM1 '-START TRACE' NORMAL COMPLETION
```

Figure 8-3   Start AUDSYSADMIN audit policy

- Start multiple audit policies by specifying multiple audit policies in the start trace AUDTPLCY keyword (Figure 8-4).

```
-START TRACE(AUDIT) AUDTPLCY(AUDEMP,AUDDEPT) DEST(GTF)
DSNW130I  -DB0B AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 04
DSNW192I  -DB0B AUDIT POLICY SUMMARY
AUDIT POLICY AUDEMP STARTED
AUDIT POLICY AUDDEPT STARTED
END AUDIT POLICY SUMMARY
DSN9022I  -DB0B DSNWVCM1 '-START TRACE' NORMAL COMPLETION
```

Figure 8-4   Start multiple audit trace policies with one start trace command

- Display the status of the audit policy that was activated in Figure 8-3. The DISPLAY command was issued with detail level 1 and 2 to provide all detail information (Figure 8-5). The DISPLAY TRACE output shows the name of the audit policy that we started and the role name for which the trace data is filtered.

```
-DISPLAY TRACE(AUDIT) AUDTPLCY(AUDSYSADMIN) DETAIL(1,2)
DSNW127I  -DB0B CURRENT TRACE ACTIVITY IS -
TNO TYPE   CLASS        DEST QUAL IFCID
04  AUDIT  *            GTF  YES
*********END OF DISPLAY TRACE SUMMARY DATA*********
DSNW143I  -DB0B CURRENT TRACE QUALIFICATIONS ARE -
DSNW152I  -DB0B BEGIN TNO 04 QUALIFICATIONS:
ROLE: DB0BSYSADM
END TNO 04 QUALIFICATIONS
DSNW185I  -DB0B BEGIN TNO 04 AUDIT POLICIES:
ACTIVE AUDIT POLICY: AUDSYSADMIN
END TNO 04 AUDIT POLICIES
DSNW148I  -DB0B ******END OF DISPLAY TRACE QUALIFICATION DATA******
DSN9022I  -DB0B DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION
```

Figure 8-5   Display audit policy AUDSYSADM

- Stop the audit trace activated in Figure 8-3 on page 160. The STOP AUDIT TRACE command specifically stops the AUDSYSADMIN audit policy (Figure 8-6).

```
-STOP    TRACE(AUDIT)  AUDTPLCY(AUDSYSADMIN    )
DSNW131I  -DB0B STOP TRACE SUCCESSFUL FOR TRACE NUMBER(S) 03
DSN9022I  -DB0B DSNWVCM1 '-STOP TRACE' NORMAL COMPLETION
```

Figure 8-6   Stop audit policy AUDSYSADMIN

As shown in Figure 8-5 on page 160 and in Figure 8-6, using the AUDTPLCY parameter is sufficient to display and stop the DB2 audit policies. You do not need to provide the trace number that was assigned by DB2 during TRACE START.

Error situations

DB2 10 introduces reason codes and messages that are related to audit policy processing. These codes and messages can assist in problem assessment and determination. When we tested our audit policy scenarios, we experienced the following common error situations:

- Start the AUDSYSADMINFAIL audit policy. Because the AUDSYSADMINFAIL audit policy does not exist, DB2 returns error reason code 00E70022 (Figure 8-7).

```
-START TRACE(AUDIT)  AUDTPLCY(AUDSYSADMINFAIL) DEST(GTF)
DSNW192I  -DB0B AUDIT POLICY SUMMARY
AUDIT POLICY AUDSYSADMINFAIL NOT STARTED, REASON 00E70022
END AUDIT POLICY SUMMARY
DSN9023I  -DB0B DSNWVCM1 '-START TRACE' ABNORMAL COMPLETION
```

Figure 8-7   Start AUDTPLCY reason code 00E70022

- Start the AUDSYSADMIN audit policy created in Figure 8-3 on page 160. Because the AUDSYSADMIN audit policy is not defined, the START command fails with reason code 00E70021 (Figure 8-8).

```
-START TRACE(AUDIT)  AUDTPLCY(AUDSYSADMIN    ) DEST(GTF)
DSNW192I  -DB0B AUDIT POLICY SUMMARY
AUDIT POLICY AUDSYSADMIN NOT STARTED, REASON 00E70021
END AUDIT POLICY SUMMARY
DSN9023I  -DB0B DSNWVCM1 '-START TRACE' ABNORMAL COMPLETION
```

Figure 8-8   Start AUDTPLCY reason code 00E70021

- Start the AUDPHONE and AUDSYSADMIN audit policies. The AUDPHONE audit policy references the PHONENO table. The START command fails with reason code 00E70024, because the table referenced in the PHONENO table does not exist. DB2 does not verify the existence of a table when you insert an audit policy into SYSIBM.SYSAUDITPOLICIES (Figure 8-9).

```
-START TRACE(AUDIT) AUDTPLCY(AUDPHONE,AUDSYSADMIN) DEST(SMF)
DSNW130I  -DB0B AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSNW192I  -DB0B AUDIT POLICY SUMMARY
AUDIT POLICY AUDSYSADMIN STARTED
AUDIT POLICY AUDPHONE NOT STARTED, REASON 00E70024
END AUDIT POLICY SUMMARY
DSN9022I  -DB0B DSNWVCM1 '-START TRACE' NORMAL COMPLETION
```

Figure 8-9 Start AUDTPLCY reason code 00E70024

Auditing START AUDIT TRACE commands

DB2 10 externalizes an IFCID 362 trace record each time an audit policy starts or fails. In case a START POLICY command fails, DB2 issues the DSNW196I warning message. For example, a START TRACE AUDTPLCY command can fail when the SYSIBM.SYSAUDITPOLICIES table row contains an invalid value in any of the category columns or if you try to start a policy that is not yet created (as shown in Figure 8-7 on page 161 and Figure 8-8 on page 161).

Figure 8-10 shows the sample report of an IFCID 362 trace record formatted by IBM Tivoli OMEGAMON® XE for DB2 Performance Expert Version V5R1 (OMEGAMON PE).

```
SYSOPR  DB0B   C66CDC2F49AA 'BLANK'      'BLANK'            'BLANK'
SYSOPR   022.AUDT 'BLANK'     00:14:05.66735526     48   1 362 START/STOP TRC NETWORKID:
'BLANK'  BL01          N/P                 AUDITPOLICY

        STATUS          : SUCCESS            TYPE           : START TRACE
     DB2 START UP   : 'BLANK'       DATABASE NAME   : 'BLANK'
      AUDIT CATEGORIES : EXECUTE
      AUDIT POLICY NAME:  AUDDEPT
      TABLE SCHEMA NAME:  DB2R5
          TABLE NAME       :  'AUDDEPT'
```

Figure 8-10 OMEGAMON PE IFCID 362 RECTRACE report

Additional information: For information about DB2 START, STOP, and DISPLAY TRACE command options, refer to *DB2 10 for z/OS Command Reference,* SC19-2972.

Changing an audit policy

You can change an existing audit policy using an SQL update while an audit trace is active for that policy. However, updating the policy using SQL does not change the audit trace criteria that was selected by DB2 during the audit trace start. To reflect policy changes in the audit trace, use the DB2 command interface to stop and start the audit policy.

You can keep track of audit policy changes by having an appropriate audit policy started. In our DB2 10 environment, we ran an update on SYSIBM.SYSAUDITPOLICIES under SYSADM authority while the audit policy AUDSYSADMIN was started. Because of this policy, DB2 wrote an IFCID 361 trace record as a result of the update statement in Example 8-2 on page 158.

Figure 8-11 shows the OMEGAMON PE record trace report that we created for IFCID 361.

```
PRIMAUTH CONNECT  INSTANCE   END_USER     WS_NAME
ORIGAUTH CORRNAME CONNTYPE   RECORD TIME  DESTNO ACE IFC DESCRIPTION
PLANNAME CORRNMBR            TCB CPU TIME            ID
-------- -------- ---------- ------------ ------ --- --- -----------
DB2R5    DB2R5X   TSO        17:36:53.164   421    3 361 AUDIT ADMIN
DSNTEP10 'BLANK'             N/P                         AUTHORITIES

        AUTHORITY TYPE   : SYSADM          AUTHID TYPE        : AUTHORIZATION ID
        PRIVILEGE CHECKED:  53 PRIVILEGE : UPDATE OBJECT TYPE : TABLE OR VIEW
        AUTHORIZATION ID :  DB2R5
        SOURCE OBJ QUALIF:  SYSIBM
        SOURCE OBJ NAME  :  SYSAUDITPOLICIES

        SQL STATEMENT:
        UPDATE SYSIBM.SYSAUDITPOLICIES SET EXECUTE = 'A'
         WHERE AUDITPOLICYNAME IN ('AUDEMP','AUDDEPT')
```

Figure 8-11 OMEGAMON PE IFCID 361 record trace

Reason codes

To handle error situations that can occur during START, STOP, and DISPLAY AUDIT TRACE commands using the new AUDTPLCY keyword, DB2 10 introduces reason codes. For details about these reason codes, refer to *DB2 10 for z/OS Codes,* GC19-2971.

Messages

DB2 10 introduces message IDs that are related to audit policy processing. These messages can assist you in problem determination and in system automation. For details about these messages for DB2 audit policies, refer to *DB2 10 for z/OS Messages,* GC19-2979.

8.1.3 Authorization

The following privileges are required to manage the policy-based auditing process:

- INSERT privilege on table SYSIBM.SYSAUDITPOLICIES for defining the policy
- TRACE privilege for activating the policy using the START AUDIT TRACE command

In DB2 10, both privileges are implicitly held by the SECADM authority, which simplifies DB2 auditing because only one authority is required for audit policy management and activation. Because the SECADM authority can grant the privilege to perform inserts on the SYSIBM.SYSAUDITPOLICIES catalog table, the management task of defining policies can be delegated to other users, for example, to users with TRACE authority. The SECADM authority has the implicit privilege to issue the start, stop, and display trace commands. Any user who has the necessary privileges to issue DB2 commands can specify the AUDTPLCY keyword on the START, STOP, and DISPLAY TRACE commands.

The following authorities implicitly hold select privilege on the SYSIBM.SYSAUDITPOLICIES catalog table:

- ACCESSCTRL
- DATAACCESS
- SYSTEM DBADM
- SQLADM
- SYSCTRL
- SYSADM

8.1.4 Policy-based SQL statement auditing for tables

DB2 10 implements policy-based SQL statement auditing of tables. The table audit capability enables you to dynamically enable SQL statement auditing for tables that do or do not have the table AUDIT clause set in their table definition.

Prior to DB2 10 for z/OS, when declaring the audit clause on specified tables, only the first statement in each unit of recovery was reflected in the audit trace. With DB2 10, it is now possible to collect each SQL statement, which can reflect a more accurate representation of events of audit interest.

When table-based SQL statement auditing is active, DB2 writes IFCID 143, 144, and 145 trace records for every SQL statement that performs changes on (SQL INSERT, UPDATE, or DELETE) or reads from (SQL SELECT) tables that are identified in the audit policy. IFCID 145 records bind time information, including the full SQL statement text and a unique SQL statement ID. The SQL statement ID is used in the IFCID 143 and IFCID 144 trace records to record any changes or reads that are performed by the SQL statement identified in the IFCID 145 trace records.

For the tables identified in the policy, DB2 writes one audit record for each SQL statement ID accessing the table. DB2 writes additional audit records when a table identified by the audit policy is read or changed using a different SQL statement ID.

You create and manage the audit policy for SQL statement auditing for tables using the interfaces that we described in 8.1.1, “Audit policies” on page 152.

DB2 10 introduces the following IFCID changes to support the new SQL statement ID based auditing for tables:

- IFCID 143 includes a unique statement identifier.
- IFCID 144 includes a unique statement identifier.
- IFCID 145 traces the entire SQL statement text and include a unique statement identifier.

8.1.5 Unique statement identifier

DB2 10 introduces a unique statement execution identifier (STMT_ID) to facilitate monitoring and tracing at the statement level. The statement ID is defined at the DB2 for z/OS server, returned to the DRDA application requester, and captured in IFCID records for both static and dynamic SQL.

For dynamic statement, the statement identifier is available when dynamic statement cache is enabled. For static statement, the statement identifier matches the STMT_ID column value in the SYSIBM.SYSPACKSTMT table. STMT_ID column is a new column in the SYSIBM.SYSPACKSTMT table.

To use the enhanced monitoring support functions, you must rebind or bind any existing pre-DB2 10 package in DB2 10 new function mode in order for the STMT_ID column to be populated and loaded into the package.

Through DRDA, the statement identifier is returned to the application requesters in addition to a 2-byte header information that includes the SQL type information (indicating a dynamic or static statement), and a 10-byte compilation time in a representation of a time stamp format with the form of `yyyymmddhhmmssnnnnnn` (the last bind time for static SQL, and the time of compilation for dynamic SQL). This information is contained in DRDA MONITORID object and is returned through the DRDA MONITORRD reply. When DRDA MONITORID is returned in the DRDA MONITORRD reply data, DB2 for z/OS server also returns a DRDA SRVNAM object along in the DRDA MONITORRD reply data providing the specific server.

The following IFCIDs are changed to externalize the new statement identifier:

- IFCID 58 (End SQL) is enhanced to return statement type, statement execution identifier, and statement level performance metrics. For related statements such as OPEN, FETCH, and CLOSE, only the IFCID 58 of the CLOSE request provides information that reflects the statistics collected on the OPEN and FETCHes as well.
- IFCID 63 and 350 (SQL Statement) are enhanced to return statement type, statement execution identifier, and the original source CCSID of the SQL statement. This new information is returned for the statement types that are candidates to be in the DB2 dynamic statement cache (SELECT, INSERT, UPDATE, DELETE, MERGE, and so on) when the dynamic statement cache is enabled.
- IFCID 124 (SQL Statement Record) is enhanced to return the statement type and statement execution identifier.
- IFCID 66 (Close Cursor) is enhanced to trace implicit close statement. A new field is added to indicate if the close statement is an explicit close statement or an implicit close statement.
- IFCID 65 (Open Cursor) is enhanced to trace the cursor attribute on the implicit commit request. A new field is added to indicate if the implicit commit cursor attribute is specified.
- IFCID 172 and IFCID 196 (Deadlock/Timeout) are enhanced to return a statement type and statement execution identifier for both dynamic and static statements when a deadlock or timeout condition is detected.
- IFCID 337 (Lock Escalation) is enhanced to return a statement type and statement execution identifier for both dynamic and static statement when a lock escalation condition is detected.

In addition, the THREAD-INFO token is extended to include statement type information, besides role and statement ID information. The affected messages are DSNL030I, DSNV436I, DSNL027I, DSNI031I, DSNT318I, DSNT375I, DSNT376I, DSNT377I, DSNT378I, DSNT771I, and DSNT772I.

Important: At publication time, the only OMEGAMON PE DB2 report that supports the new SQL statement ID based auditing is the DB2PM trace report. This identifier is not currently supported in either the AUDIT report set or the FILE command with the AUDIT report set.

For dynamic SQL statements, the unique statement identifier (STMT_ID) is derived from the dynamic statement cache, and for embedded static SQL statements, the STMT_ID is derived from the SYSIBM.SYSPACKSTMT.STMT_ID catalog table column.

You cannot use the table-based auditing with tables that are defined in simple table spaces.

Audit policy for table-based SQL auditing

If you want to activate table-based auditing for SQL statements, you need to create an audit policy for the EXECUTE category. Additionally, you need to provide further policy information to specify the object type, object name, and the schema name that you want to audit. The following SYSIBM.SYSAUDITPOLICIES columns are relevant to the audit category EXECUTE:

- AUDITPOLICYNAME
- OBJECTSCHEMA
- OBJECTNAME
- OBJECTTYPE
- EXECUTE

For more information about these columns, refer to Table 8-2 on page 154.

Sample scenario

We prepare a sample scenario with two sample tables, an embedded SQL program, and a small dynamic SQL workload to illustrate table-based auditing. The sample scenario contained the following objects:

- Tables DB2R5.AUDEMP and DB2R5.AUDDEPT
- Dynamic SQL

  The dynamic SQL workload shown in Example 8-3 executes three different SQL statements within one unit of work. The SQL statements access the same table. We therefore see three IFCID 143 records within one unit of work, one for each distinct SQL DML statement.

  Example 8-3 AUD dynamic SQL for auditing

```
INSERT INTO DB2R5.AUDEMP
 VALUES (   '300000' , 'JOSEF' , ' ' , 'KLITSCH' , 'X00' , '1234' ,
'1998-08-29','ITJOB',42,'M','1958-09-13',99.99,99.99,578);
INSERT INTO DB2R5.AUDEMP
 VALUES (   '300001' , 'JOSEF' , ' ' , 'KLITSCH' , 'X00' , '1234' ,
'1998-08-29','ITJOB',42,'M','1958-09-13',99.99,99.99,578);
DELETE FROM DB2R5.AUDEMP WHERE EMPNO >= 300000';
COMMIT  ;
```

- Static SQL COBOL program: AUDSQL

  The COBOL program AUDSQL executes the static SQL statements shown in Example 8-4. The program contains two different SELECT statements, and each statement is executed three times within each unit of work. Therefore, we see two IFCID 144 records for each unit of work, one for each distinct SQL statement.

  Example 8-4 AUD SQL static SQL for auditing

```
PERFORM 3 TIMES
   PERFORM 3 TIMES
         EXEC SQL
              SELECT COUNT(*) INTO :V-EMP FROM DB2R5.AUDDEPT
         END-EXEC
         EXEC SQL
              SELECT COUNT(*) INTO :V-EMP FROM DB2R5.AUDEMP
         END-EXEC
   END-PERFORM
   EXEC SQL COMMIT END-EXEC
END-PERFORM.
```

Creating an EXECUTE category audit policy

Within this activity, we execute the SQL statements shown in Example 8-5 to create an audit policy to audit any SQL activity on tables DB2R5.AUD%. We set OBJECTNAME to `AUD%`, which causes DB2 to use a like predicate when determining the tables to be audited. Example 8-5 inserts an audit policy to audit tables names that begin with `AUD% within schema DB2R5`.

Example 8-5 Create SQL statement auditing policy

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
VALUES('AUDTABLES','DB2R5','''AUD%''','T','A');
```

Starting the audit policy

Under an authority with TRACE privilege, we successfully start the audit policy for table based SQL auditing (Example 8-6).

Example 8-6 Start SQL statement auditing policy

```
-START TRACE(AUDIT) AUDTPLCY(AUDTABLES) DEST(GTF)
DSNW130I  -DB0B AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSNW192I  -DB0B AUDIT POLICY SUMMARY
AUDIT POLICY AUDTABLES STARTED
END AUDIT POLICY SUMMARY
DSN9022I  -DB0B DSNWVCM1 '-START TRACE' NORMAL COMPLETION
```

Displaying the audit policy status

If you are not sure an audit policy is started, you can use the DISPLAY TRACE command with the AUDTPLCY parameter to verify that audit policy started. To run the DISPLAY TRACE command, you must have the DISPLAY TRACE privilege, which is included in the SYSTEM DBADM, the SYSOPR, the SYSCTRL, the SYSADM, and in the SECADM authority. Example 8-7 shows the output of the DISPLAY TRACE command with the AUDTPLCY parameter.

Example 8-7 Display SQL statement audit policy

```
-DIS   TRACE(AUDIT) AUDTPLCY(AUDTABLES) DETAIL(1,2)
DSNW127I  -DB0B CURRENT TRACE ACTIVITY IS -
TNO TYPE   CLASS        DEST QUAL IFCID
03  AUDIT  *            GTF  NO
*********END OF DISPLAY TRACE SUMMARY DATA*********
DSNW143I  -DB0B CURRENT TRACE QUALIFICATIONS ARE -
DSNW152I  -DB0B BEGIN TNO 03 QUALIFICATIONS:
NO QUALIFICATIONS
END TNO 03 QUALIFICATIONS
DSNW185I  -DB0B BEGIN TNO 03 AUDIT POLICIES:
ACTIVE AUDIT POLICY: AUDTABLES
END TNO 03 AUDIT POLICIES
DSNW148I  -DB0B ******END OF DISPLAY TRACE QUALIFICATION DATA******
DSN9022I  -DB0B DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION
```

Stopping the audit policy

After successful SQL workload execution, we run the command in Example 8-8 to stop the audit policy. The command requires TRACE or SECADM authority.

Example 8-8 Stop SQL statement audit policy

```
-STOP  TRACE(AUDIT) AUDTPLCY(AUDTABLES)
DSNW131I  -DB0B STOP TRACE SUCCESSFUL FOR TRACE NUMBER(S) 03
DSN9022I  -DB0B DSNWVCM1 '-STOP TRACE' NORMAL COMPLETION
```

Audit policy trace and increased granularity of event collection

In terms of performance, when using the new audit policies, you need to be aware of which IFCIDs are collected and when, to evaluate the impact of auditing. With audit policies, you can collect *all* the DML accesses to a table. The audit trace instead only records the *first* access within the unit of recovery, which may constitute an incomplete auditing for some institutions. Audit policies also add IFCIDs that were not included with audit traces to some audit categories.

Sample auditing test comparing a trace to audit policy

Our test uses a simple application that issues 16 separate SELECT statements against a table. We then compare the differences in setup for audit trace and audit policy. Note that the SMF data sets used during the tests have been SWITCHed during the tests, but each data set contains more than one test result, so we generated the selected output by using the FROM/TO clause of OMEGAMON PE reporter function when showing the results.

Example 8-9 shows how to start an audit trace using the START TRACE command. In our case, the table we are going to audit is defined with AUDIT ALL in its DDL. If you change the AUDIT specification using the ALTER TABLE statement, the packages related to that table will be invalidated and the auto BIND will take place the next time you execute the application. You should REBIND the packages explicitly if you have ABIND set to NO, or the execution fails.

Example 8-9 START TRACE for auditing DML access to a table with audit trace

```
-DB0A STA TRACE(AUDIT) CLASS(4,5,6) DEST(SMF)
DSNW130I  -DB0A AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSN9022I  -DB0A DSNWVCM1 '-STA TRACE' NORMAL COMPLETION
```

When you run a OMEGAMON PE reporter to report the trace after a single execution of the application, you get the IFCID frequency distribution log shown in Example 8-10. In this report, you can see the number of IFCID records counted within the input SMF data sets and also the number of records processed for the report. For our static SQL application accessing the table, one IFCID 144 record was processed, or recorded, with the audit trace report, even though 16 SELECTs have been issued. This is due to the limitation of the audit trace, which only records the first access to the table within the unit of recovery.

Example 8-10 OMEGAMON IFCID frequency distribution log for audit trace with static SQL statements

```
1   LOCATION: DB0A                               OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)              PAGE: 1
       GROUP: N/P                               IFCID FREQUENCY DISTRIBUTION LOG                     RUN DATE: 11-03-10 17:06:51.97
      MEMBER: N/P
   SUBSYSTEM: DB0A                                                                                 ACTUAL FROM: 11-03-10 00:38:25.43
 DB2 VERSION: V10                                                                                          TO: 11-03-10 01:21:00.00
0             INPUT       INPUT     PROCESSED     PROCESSED                   INPUT       INPUT     PROCESSED     PROCESSED
    IFCID     COUNT    PCT OF TOTAL   COUNT     PCT OF TOTAL         IFCID     COUNT    PCT OF TOTAL   COUNT     PCT OF TOTAL
   -------  ----------  ------------  ----------  ------------       -------  ----------  ------------  ----------  ------------
       1          43      14.19%            0       0.00%             144          23       7.59%            1       2.17%
       2          43      14.19%            0       0.00%             145           6       1.98%            0       0.00%
       3           5       1.65%            1       2.17%             202          43      14.19%           43      93.47%
```

```
        4           2      0.66%           0      0.00%             225           43     14.19%            0      0.00%
        5           3      0.99%           0      0.00%             239            5      1.65%            1      2.17%
      105          43     14.19%           0      0.00%             362            1      0.33%            0      0.00%
      106          43     14.19%           0      0.00%
0      TOTAL INPUT TRACE RECORDS  =              303
       TOTAL PROCESSED TRACE RECORDS =            46
```

We now execute the same application with the audit policy. First, we populate the SYSIBM.SYSAUDITPOLICIES to collect the EXECUTE *audit category;* this category collects all DML accesses (Example 8-11).

Example 8-11 Audit policy creation

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
VALUES('AUDTB1','DB2R6','TABLE1','T','A');
```

We then start the audit policy, as shown in Example 8-12. Unlike the audit trace where you need to REBIND after ALTERing your table with the AUDIT specification, in this case auditing starts after you start the audit policies without the need to REBIND the package.

Example 8-12 START TRACE for auditing DML access to a table with audit policies

```
-DB0A STA TRACE(AUDIT) AUDTPLCY(AUDTBS)
DSNW130I  -DB0A AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSNW192I  -DB0A AUDIT POLICY SUMMARY 778
AUDIT POLICY AUDTBS STARTED
END AUDIT POLICY SUMMARY
DSN9022I  -DB0A DSNWVCM1 '-STA TRACE' NORMAL COMPLETION
```

Example 8-13 shows the IFCID frequency distribution log with OMEGAMON PE. You can see 16 IFCID 144 records processed for the report, which confirms that an audit record was written for each SELECT statement accessing a table within the application.

Example 8-13 OMEGAMON IFCID frequency distribution log for audit policies with static SQL statements

```
1   LOCATION: DB0A                                  OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)             PAGE: 1
       GROUP: N/P                                  IFCID FREQUENCY DISTRIBUTION LOG                      RUN DATE: 11-03-10 17:04:12.58
      MEMBER: N/P
   SUBSYSTEM: DB0A                                                                                     ACTUAL FROM: 11-03-10 00:38:25.43
 DB2 VERSION: V10                                                                                               TO: 11-03-10 01:21:00.00
0               INPUT        INPUT      PROCESSED     PROCESSED                       INPUT        INPUT      PROCESSED     PROCESSED
    IFCID       COUNT     PCT OF TOTAL    COUNT     PCT OF TOTAL          IFCID       COUNT     PCT OF TOTAL    COUNT     PCT OF TOTAL
    -------  ----------  ------------  ----------  ------------         -------  ----------  ------------  ----------  ------------
        1          43     14.19%           0      0.00%             144           23      7.59%           16     26.22%
        2          43     14.19%           0      0.00%             145            6      1.98%            0      0.00%
        3           5      1.65%           1      1.63%             202           43     14.19%           43     70.49%
        4           2      0.66%           0      0.00%             225           43     14.19%            0      0.00%
        5           3      0.99%           0      0.00%             239            5      1.65%            1      1.63%
      105          43     14.19%           0      0.00%             362            1      0.33%            0      0.00%
      106          43     14.19%           0      0.00%
0      TOTAL INPUT TRACE RECORDS  =              303
       TOTAL PROCESSED TRACE RECORDS =            61
```

It is obvious that you get better auditing capability with audit categories, but, depending on the average of DML statements executed per transaction when accessing the same table within the same unit of recovery, additional audit trace records are created when you start using audit policies instead of the audit trace.

Compressing your audit records

When audit policies are used instead of audit traces, you are expected to get more trace records. This is due to several changes to include additional information. The volume of data directed to SMF can be immense. DB2 10 introduces a new DSNZPARM parameter, SMFCOMP, which directs DB2 to request compression for trace records that are sent to SMF. Trace data to GTF and OPX is not compressed. SMFCOMP is specified on the installation panel DSNTIPN in the field COMPRESS SMF RECS. The default value is OFF.

If compression is enabled, SMF data is compressed such that everything after the SMF header (SM100END, SM101END, or SM102END) is compressed with z/OS compression service CSRCESRV. A compressed record is identified by a bit in the SMF100, 101, and 102 headers. The trade-off for this function is SMF volume versus an increase in CPU to compress and expand the records.

Performance measurements show a minimal impact of up to 1% with accounting class 1, 2, 3, 7, 8, and 10 active. The disk savings for DB2 SMF data set can be significant with a compression rate of 60% to 80%. Our results showed statistics classes 1, 3, 4, 5, and 6 with a compression rate around 60%.

If you use your own trace formatter, you need to call the z/OS compression service, which is turned off by default, to decompress the data. APAR PM27872 provides you with decompression routine DSNTSMFD and a sample JCL, DSNTEJDS, to execute it. DSNTSMFD takes SMF data as an input to decompress. DSNTSMFD gives you an output message of how much your data was compressed and the percentage saved by the compression.

We show how the audit trace can take advantage of the SMF compress function by setting SMFCOMP to ON and using an application that executes 20,000 simple SELECT statements against a table. The trace generates 20,000 IFCID 144 records. The tool OMEGAMON PE can automatically process compressed SMF records.

If you are using SMF as an input to your own application, instead of OMEGAMON PE, you can use the DB2 provided sample program DSNTSMFD to decompress compressed DB2 trace data. APAR PM27872 has provided DSNTSMFD. This program also gives the percentage of space saved using SMF compression.

Figure 8-12 shows the output from the decompression sample program DSNTSMFD.

```
 *** DSNTSMFD *** STARTING     2011/03/11         16:49:35

-------------------------------------------------------------------------------
 Total records read:...............................      20370
   Total DB2 records read:.........................      20293
     Total DB2 compressed records read:............      20090
     Total DB2 compressed records decompressed:....      20090
   Total non-DB2 records read:.....................         77

 Aggregate size of all input records:..............    4156941          3M
   Aggregate size of all input DB2 records:........    4131027          3M
     Aggregate size of all DB2 compressed records:...  3767659          3M
   Aggregate size of all output DB2 records:.......    6587780          6M
     Aggregate size of all DB2 expanded records:.....  6224412          5M
   Aggregate size of all non-DB2 input records:......    25914         25K

     Percentage saved using compression.............        39%

 Details by DB2 subsystem
     Subsystem ID:  DB0A
```

Figure 8-12 DSNTSMFD output for audit records - IFCID 144

We see that 20,032 out of the 20,096 DB2 trace records within SMF are IFCID 144 records. Our test application only shows 39% in savings because more than 99% of the DB2 trace records are IFCID 144 records. In general, you are expected to have better compression ratio because you get better compression ratio with accounting trace, statistics trace, and most other audit trace records.

Reports for SQL auditing

In this section, we describe how we used OMEGAMON PE to format the IFCID traces that we collected within our sample scenario.

Figure 8-13 shows the OMEGAMON PE record trace report that we created for our static SQL sample workload. The report shows six IFCID 144 trace records (two for each DB2 unit of work) that DB2 collected because of the SQL workload shown in Example 8-4 on page 166.

```
DB2R5    BATCH    C681E3C79388 'BLANK'         'BLANK'                       'BLANK'
DB2R5    DB2R5R06 TSO         17:39:53.41335706      9   1 361 AUDIT ADMIN  NETWORKID:  USIBMSC
AUDSQL   'BLANK'              N/P                               AUTHORITIES

AUTHORITY TYPE   : SYSADM       AUTHID TYPE : AUTHORIZATION ID
PRIVILEGE CHECKED:         64   PRIVILEGE   : EXECUTE           OBJECT TYPE : APPLICATION PLAN
AUTHORIZATION ID :  DB2R5
SOURCE OBJ NAME  :  AUDSQL

SQL STATEMENT: 'BLANK'
.................................................................................................
17:39:53.41637954  10   1 144 AUDIT FIRST    'BLANK'
N/P                           READ           NETWORKID:  USIBMSC   LUNAME:  SCPDBOB   LUWSEQ:  1
                                             DATABASE: 285           LOGRBA: X'000000000000'
                                             PAGE SET: 222                   TABLE OBID:   223
                                             STMT ID :                24682
17:39:53.47192559  11   1 144 AUDIT FIRST    'BLANK'
N/P                           READ           NETWORKID:  USIBMSC   LUNAME:  SCPDBOB   LUWSEQ: 1
                                             DATABASE: 285           LOGRBA: X'000000000000'
                                             PAGE SET: 219                   TABLE OBID:  220
                                             STMT ID :                24683
17:39:53.50356837  12   1 144 AUDIT FIRST    'BLANK'
N/P                           READ           NETWORKID:  USIBMSC   LUNAME:  SCPDBOB   LUWSEQ: 2
                                             DATABASE: 285           LOGRBA: X'000000000000'
                                             PAGE SET: 222                   TABLE OBID:  223
                                             STMT ID :                24682
17:39:53.50384456  13   1 144 AUDIT FIRST    'BLANK'
N/P                           READ           NETWORKID:  USIBMSC   LUNAME:  SCPDBOB   LUWSEQ: 2
                                             DATABASE: 285           LOGRBA: X'000000000000'
                                             PAGE SET: 219                   TABLE OBID:  220
                                             STMT ID :                24683
17:39:53.50402484  14   1 144 AUDIT FIRST    'BLANK'
N/P                           READ           NETWORKID:  USIBMSC   LUNAME:  SCPDBOB   LUWSEQ: 3
                                             DATABASE: 285           LOGRBA: X'000000000000'
                                             PAGE SET: 222                   TABLE OBID:  223
                                             STMT ID :                24682
17:39:53.50428398  15   1 144 AUDIT FIRST    'BLANK'
N/P                           READ           NETWORKID:  USIBMSC   LUNAME:  SCPDBOB   LUWSEQ: 3
                                             DATABASE: 285           LOGRBA: X'000000000000'
                                             PAGE SET: 219                   TABLE OBID:  220
                                             STMT ID :                24683
```

Figure 8-13 OMEGAMON PE record trace for static SQL

8.1.6 Auditing authorities with audit policies

DB2 10 for z/OS provides additional system authorities and privileges. These authorities and privileges are designed to help business to comply with government regulations and to simplify the management of authorities. DB2 10 introduces the concepts of separation of duties and least privilege to address these needs.

Separation of duties provides the ability for administrative authorities to be divided across individuals without overlapping responsibilities so that one user does not possess unlimited authority (for example, SYSADM). For example, in DB2 10, a SYSTEM DBADM authority provides the ability to manage any table without having data access to the table.

In addition, a SECADM authority provides the ability to manage access to tables in DB2 while not holding the privilege to create, alter, drop, or access a table. Furthermore, a DATAACESS authority can access all user tables without being able to manage security or database objects. You can use these authorities to minimize the need for the SYSADM authority in day-to-day operations by delegating security administration, system related database administration, and database access duties onto the SECADM, SYSTEM DBADM, and DATAACCESS authorities. To go even further, you can separate security administration from the SYSADM authority so that SYSADM can no longer perform security-related tasks.

To demonstrate the appropriate use of these administrative authorities, DB2 10 for z/OS provides a way to implement auditing whenever these authorities are used, and to control the audit event collection through the use of audit policies. The following sections provide several sample scenarios showing how to audit administrative authorities.

Auditing the SECADM authority

You can use a category SECMAINT audit policy to keep track of grants and revokes performed by SECADM authorities. In our DB2 10 environment, we used the SQL statement shown in Example 8-14 to create an audit policy to audit all grants and revokes performed by SECADM authorities.

Example 8-14 SECMAINT audit policy - Grant - Revoke auditing

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
   (AUDITPOLICYNAME, SECMAINT) VALUES('AUDSECMAINT','A');
```

We then started the audit policy and used the SECADM authority authorization ID DB0BSECA to grant the DATAACCESS privilege to authorization ID DB0BDA (Example 8-15).

Example 8-15 SECMAINT audit policy - Grant DATAACESS SQL

```
SET CURRENT SQLID = 'DB0BSECA';
GRANT  DATAACCESS ON SYSTEM TO   DB0BDA;
```

As a result, an IFCID 141 audit trace record with an audit reason of SECADM was created by DB2. The OMEGAMON PE formatted audit trace for this trace record is shown in Figure 8-14.

```
TIMESTAMP   TYPE                                   DETAIL
----------- -------- ------------------------------------------------------------------
18:32:11.91 AUTHCNTL GRANTOR: DB0BSECA   OWNER TYPE: PRIM/SECOND AUTHID
                                         REASON: SECADM               SQLCODE:     0
                     OBJECT TYPE: USERAUTH
                     TEXT: GRANT DATAACCESS ON SYSTEM TO DB0BDA
```

Figure 8-14 OMEGAMON PE category SECMAINT audit report

Auditing the SYSTEM DBADM authority

You can use a category DBADMIN audit policy to keep track of activities that are performed by SYSTEM DBADM authorities. In our DB2 10 environment, we used the SQL statement shown in Example 8-16 to create an audit policy to audit all database administrative tasks performed by SYSTEM DBADM authorities. We set the DBADMIN column to “*” to audit all authorities.

Example 8-16 DBADMIN audit policy - SYSTEM DBADM auditing

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
   (AUDITPOLICYNAME, DBADMIN) VALUES('AUDDBADMIN','*');
```

We then started the audit policy and used the SYSTEM DBADM authority authorization ID DB0BSDBA to create a plan table in schema DB0BEXPLAIN (Example 8-17).

Example 8-17 DBADMIN audit policy - Create table by SYSTEM DBADM

```
SET CURRENT SQLID = 'DB0BSDBA';
CREATE TABLE DB0BEXPLAIN.PLAN_TABLE LIKE DB2R5.PLAN_TABLE;
```

As a result, DB2 created an IFCID 361 audit trace record for authorization type SYSDBADM. Figure 8-15 shows the OMEGAMON PE formatted audit trace for this trace record.

```
AUTHCNTL AUTH TYPE:         SYSDBADM
         PRIV CHECKED:      CREATE TABLE                     OBJECT TYPE: DATABASE
         AUTHID:            DB0BSDBA
         SOURCE OBJECT
          QUALIFIER:        SYSIBM
          NAME:             DSNDB04
         TARGET OBJECT
          QUALIFIER:        DB0BEXPLAIN
          NAME:             PLAN_TABLE
         TEXT:              CREATE TABLE DB0BEXPLAIN.PLAN_TABLE LIKE DB2R5.PLAN_TABLE
```

Figure 8-15 OMEGAMON PE auth type SYSDBADM audit report

Auditing the DATAACCESS authority

You can use a category DBADMIN audit policy to keep track of activities that are performed by DATAACCESS authorities. DATAACCESS allows the user to access data in all user tables, views, and materialized query tables in a DB2 subsystem and allows the user to execute plans, packages, functions, and procedures.

In our DB2 10 environment, we used the SQL statement shown in Example 8-18 to create an audit policy to audit all access and execute tasks performed by DATAACCESS authorities. We set the DBADMIN column to T to audit only DATAACCESS authorities.

Example 8-18 DBADMIN audit policy - DATAACCESS auditing

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
   (AUDITPOLICYNAME, DBADMIN) VALUES('AUDDBADMIN_T','T');
```

We then started the audit policy and used the DATAACCESS authority authorization ID DB0BDA to query plan table DB2R5.PLAN_TABLE (Example 8-19).

Example 8-19 DBADMIN audit policy - Query a table by DATAACCESS authority

```
SET CURRENT SQLID = 'DB0BDA';
SELECT * FROM DB2R5.PLAN_TABLE ;
```

As a result, DB2 created an IFCID 361 audit trace record for authorization type DATAACCESS. Figure 8-16 shows the OMEGAMON PE formatted audit trace for this trace record.

```
 TYPE                                DETAIL
-------- --------------------------------------------------------------------------------
AUTHCNTL AUTH TYPE:          DATAACCS
         PRIV CHECKED:       SELECT                           OBJECT TYPE:       TAB/VIEW
         AUTHID:             DB0BDA
         SOURCE OBJECT
          QUALIFIER:         DB2R5
          NAME:              PLAN_TABLE
         TARGET OBJECT
          QUALIFIER:         N/P
          NAME:              N/P
         TEXT:               SELECT * FROM DB2R5.PLAN_TABLE
```

Figure 8-16 OMEGAMON PE auth type DATAACCESS audit report

9

RACF and DB2

In this chapter, we provide an introduction to the two alternatives that DB2 for z/OS offers to protect its resources in general: DB2-managed security and RACF-managed security. Resource Access Control Facility (RACF) is the IBM software security system that provides access control and auditing functionality for z/OS (and VM). RACF is part of the Security Server for z/OS.

Native DB2 authorization uses the grant and revoke statements to keep the access privilege information in the DB2 catalog. Checking access means checking the DB2 catalog.

DB2 based access control uses tight integration with DB2 and database integrity techniques to provide robust security. There is not a security rule granted without a related object in most situations. However, for some customers and security needs, these techniques do not provide sufficient access control. If you want the database administrators to manage security, and integration with DB2 is important, native DB2 based authorization may be a good choice.

If you want security administrators to manage security, integration with the security server and the ability to have separate security and database administration with a RACF implementation might be more important. There are significant policy and people implications when using RACF access control. The change to RACF access control causes organizational roles and DB2 authorities to change.

Converting to RACF from DB2 security is not a completely compatible change[1]. Even with a RACF based security implementation, there are some situations where DB2 access control must be used. However, if you want a security group to define authorization from a centralized security control point, RACF access control could be a good fit. However, remember that the security and implementation team requires both DB2 and RACF knowledge for a successful security implementation.

This chapter contains the following topics:

- Authorization IDs for accessing data within DB2
- DB2 managed security
- RACF managed security

[1] Chapter 10, "Special considerations" in *DB2 10 for z/OS RACF Access Control Module Guide*, SC19-2982 lists the instances where the RACF authorization checking done by the RACF access control module is different from the authorization checking done by DB2.

9.1 Authorization IDs for accessing data within DB2

Data access in a local DB2 system can be from a batch program, from a user on an interactive terminal session, or from a CICS or IMS transaction. For the purposes of security, DB2 uses the term *process* to represent all forms of access to data. Every process that connects to or signs on to DB2 is represented by a set of one or more authorization IDs. When authorization IDs are assigned, every process receives at least one primary authorization ID and one or more secondary IDs, and one of those IDs is designated as the current SQL ID. Also, from within trusted context, a role might be assigned.

As shown in Figure 9-1, DB2 provides different ways for you to control access from all but the data set protection route.

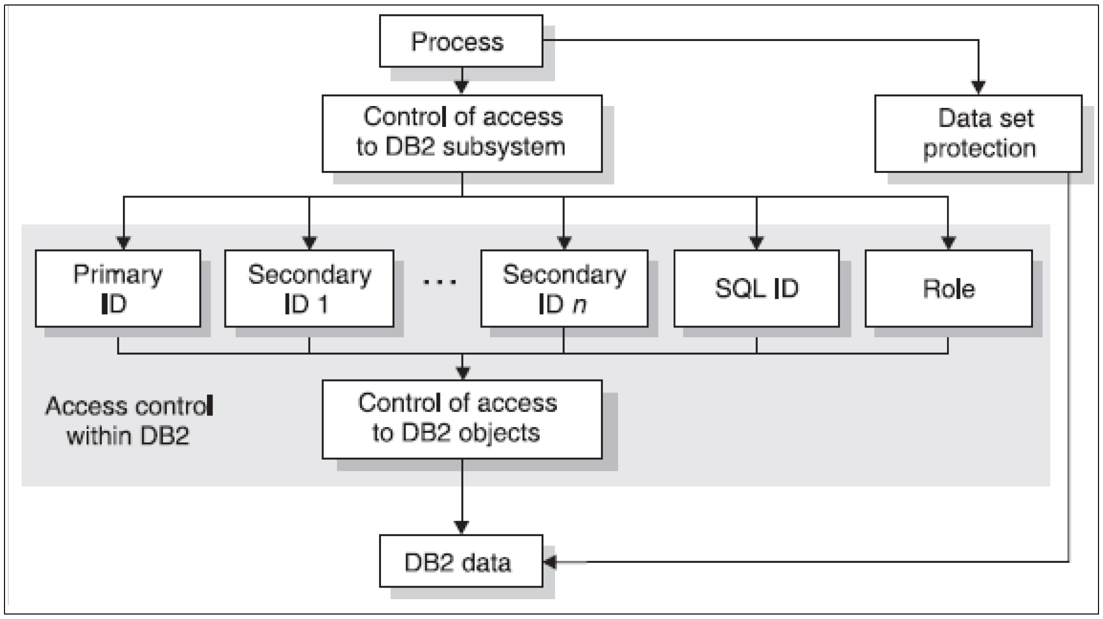

Process
Control of access to DB2 subsystem
Data set protection
Primary ID
Secondary ID 1
. . .
Secondary ID n
SQL ID
Role
Access control within DB2
Control of access to DB2 objects
DB2 data

Figure 9-1 Data access control

Note the following definitions:

Primary authorization ID	Generally, the primary authorization ID identifies a process. For example, statistics and performance trace records use a primary authorization ID to identify a process.
Secondary authorization ID	A secondary authorization ID is optional. It can hold additional privileges that are available to the process. For example, a secondary authorization ID can be a RACF group ID.
SQL ID	An SQL ID holds the privileges that are exercised when certain dynamic SQL statements are issued. The SQL ID can be set equal to the primary ID or any secondary ID.

Role	A role is available within a trusted context. You can define a role and assign it to authorization IDs in a trusted context. When associated with a role and using the trusted connection, an authorization ID inherits all the privileges granted to that role. The role, together with the trusted context, are database objects introduced in DB2 9. We describe them in Chapter 4, "Roles and trusted contexts" on page 69.

Different local processes enter the DB2 access control procedure at different points, depending on the environment on which they originate.

Processes that go through *connection* processes only include:

- Requests that originate in the TSO foreground and background, including online utilities and requests through the call attachment facility
- JES-initiated jobs
- Requests through started task control address spaces as a result of the MVS START command

Processes that go through *connection* processes and that later on can go through the *sign-on* exit include:

- The overall connection between a CICS region and DB2, which is created by the CICS DB2 attachment facility
- The IMS control region
- DL/I batch
- Requests through the Resource Recovery Services attachment facility (RRSAF)

Processes that go through *sign-on processing* are:

- Requests from IMS-dependent regions, including MPP, BMP, and Fast Path
- CICS transactions that acquire a thread into DB2

During connection processing and sign-on processing, DB2 sets the primary and secondary authorization IDs for the process to use in the DB2 address space. By default, DB2 uses the authorization IDs that the process has provided. Next to default procedures, authorization IDs can also be assigned to those processes by user-written exit routines.

DB2 has two exit points for authorization routines, one in connection processing and one in sign-on processing. Both exit points perform crucial steps in the assignment of values to primary IDs, secondary IDs, and SQL IDs. Optionally, a default role or a user specific role may also be assigned. DB2 has a default connection exit routine and a default sign-on exit routine. You can replace these routines with your own exit routines.

A sample connection exit routine and sign-on exit routine are supplied with DB2. The name of the connection authorization exit routine is DSN3@ATH, and the name of the sign-on exit routine is DSN3@SGN.

For a detailed description about the usage of these exits, see Appendix A, "Connection routines and sign-on routines", in *DB2 10 for z/OS Administration Guide,* SC19-2968.

9.1.1 Processing connections

A connection request, local or remote, makes a new connection to DB2. It does not reuse an application plan that is already allocated. By default, DB2 uses the authorization IDs that the process has provided. For a TSO user, this is the TSO logon ID. For batch programs, CICS, and IMS control region, this is the user ID of the MVS region.

During connection processing, DB2 calls RACF to check whether the ID is authorized to use:

- The DB2 resource class DSNR
- The DB2 subsystem
- The requested connection type

If RACF is active and has verified the RACF user ID, DB2 runs the connection exit routine. Running with the default DB2-provided exit routine makes sure that the default user ID is the primary authorization ID, no secondary IDs exist, and the SQL ID is the same as the primary ID.

If you want to use DB2 secondary authorization IDs, you must replace the default connection exit routine. The connection authorization exit routine must be named DSN3@ATH. DB2 installation job DSNTIJEX in SDSNSAMP provides a step to replace the default connection exit with a sample connection exit routine, of which you can find the source in the SDSNSAMP library in member DSN3SATH. Basically, the sample connection exit routine sets the DB2 primary ID and the SQL ID in the same way as the default routine.

The setting of the secondary authorization ID depends on RACF options. There can be no secondary authorization ID at all, it can be one single ID, the default connected group name, or there can be a list of secondary authorization IDs equal to the list of group names to which the RACF user ID is connected.

If you choose to use the RACF list of groups to populate the secondary auth ID list, you should ensure that there are no conflicts between the installations RACF groups and the permissions in the DB2 catalog tables.

For more details about setting the different authorization IDs during connection processing, see Appendix A, “Connection routines and sign-on routines” in *DB2 10 for z/OS Administration Guide,* SC19-2968.

Note: The space allocated for the secondary ID list is only large enough to contain the maximum number of authorization IDs which is currently 1012. If your exit allows for more than 1012 secondary authorization IDs, DB2 abends and storage overlays can occur.

As a final step of connection processing, DB2 determines whether the connection is to be established as trusted. If a trusted context with a system authorization ID matching the primary authorization ID exists, and the attributes of the trusted context match those of the connection request, the connection is established as trusted.

9.1.2 Processing sign-ons

For requests from IMS-dependent regions, CICS transaction subtasks, or RRS connections, the initial primary ID is not obtained until just before allocating a plan for a transaction. A new sign-on request can run the same plan without deallocating the plan and reallocating it. Nevertheless, the new sign-on request can change the primary ID. Unlike connection processing, sign-on processing does not check the RACF user ID of the address space.

First, DB2 determines the initial primary ID as follows:

- For CICS transactions, the ID that is used as the primary authorization ID is determined by attributes in the DB2CONN or DB2ENTRY definitions, depending on the thread type.
- For IMS sign-ons from message-driven regions, if the user has signed on, the initial primary authorization ID is the user's sign-on ID. IMS passes to DB2 the IMS sign-on ID and the associated RACF connected group name, if one exists. If the user has not signed on, the primary ID is the LTERM name, or if that is not available, the PSB name.
- For a batch-oriented region, the primary ID is the value of the USER parameter on the job statement, if that is available. If that is not available, the primary ID is the program's PSB name.

Then, DB2 runs the sign-on exit routine. Using the IBM-supplied default sign-on exit routine makes sure that the initial primary authorization ID remains the primary ID, the SQL ID is set equal to the primary ID, and no secondary IDs exist.

As with connection processing described earlier, if you want the primary authorization ID to be associated with DB2 secondary authorization IDs, you must replace the default sign-on exit routine. If you want to use RACF group names as DB2 secondary IDs, the easiest method is to use the IBM-supplied sample routine. Job DSNTIJEX in SDSNSAMP allows you to replace the default sign-on exit with the sample exit, also provided in source code as member DSN3SSGN.

The sample sign-on routine sets the initial primary authorization ID unchanged as the DB2 primary ID, and the SQL ID is made equal to the DB2 primary ID. If RACF is not active, no secondary IDs exist. If RACF is active but its list of groups option is not active, one secondary ID exists and this is the name passed by CICS or by IMS. If RACF is active and you selected the option for a list of groups, the routine sets the list of DB2 secondary IDs to the list of group names to which the RACF user ID is connected. The list of group names includes the default connected group name.

For instructions about how to prepare the sample exits, or to write your own exit routine, refer to Appendix A, “Connection routines and sign-on routines”, in *DB2 10 for z/OS Administration Guide*, SC19-2968.

As a final step of sign-on processing, DB2 checks to see whether the request is from a trusted connection performing a switch user, and if so, whether the primary authorization ID is permitted to switch. If the primary authorization ID is not allowed to switch, the connection is terminated.

9.2 DB2 managed security

Resource access control can be managed through a DB2 built-in mechanism. Two SQL statements are used to provide security control: GRANT and REVOKE. For example, grants control the use of buffer pools, utilities, storage groups, DB2 commands, and so on. Grants are recorded in the DB2 catalog.

Every process that connects to or signs on to DB2 is represented by one ID, called the primary authorization ID. All other IDs are secondary authorization IDs. One ID, either primary or secondary, is designated as the CURRENT SQLID. The CURRENT SQLID specifies the SQL authorization ID of the process. Additionally, a role can be assigned to a user if the connection is made through a trusted context.

The role is a relatively new object introduced in DB2 9 for z/OS. It is a database entity, available only in a trusted context, that groups together one or more privileges. A role can own database objects, which helps eliminate the need for individual users to own and control database objects. You can assign a role to an individual user or a group of users by defining a trusted context.

DB2 controls access to its objects by a set of privileges. Each privilege allows an action on some object. The GRANT statement grants privileges to authorization IDs or roles. Privileges can be explicit and implicit.

Explicit privileges have names and are held as the result of GRANT and REVOKE statements. There is a set of specific privileges for each type of DB2 object.

As an example, the privileges for a table are:

- ALTER: Change the table definition.
- DELETE: Delete rows in the table.
- INDEX: Create an index on the table.
- INSERT: Insert rows into the table.
- REFERENCES: Add or drop a referential constraint referring to the table.
- SELECT: Retrieve data from the table.
- TRIGGER: Define a trigger on the table.
- UPDATE: Change the contents of a specific column.

A privilege can be explicit, which means that it has a name and is held as the result of an SQL GRANT statement. Explicit privileges provide detailed control. Examples are:

```
GRANT SELECT ON TABLE SYSIBM.SYSDATABASE TO USER1
GRANT SELECT ON TABLE SYSIBM.SYSUSERAUTH TO PUBLIC
```

The first statement gives USER1 the authority to select from the SYSIBM.SYSDATABASE table. The second statement allows all users to select rows from SYSIBM.SYSUSERAUTH. In the example, USER1 can be anything: a TSO user ID, a RACF group representing a list of users, a role, or even just a short character string.

Sets of privileges are grouped into administrative authorities. These authorities form a hierarchy. Each hierarchy includes a specific group of privileges. The administrative authorities fall into the categories of system, database, and collection authorities. The highest ranking administrative authority is SYSADM. Each level of authority includes the privileges of all lower-ranking authorities. Examples of authorities can include system authorities such as SYSADM or database authorities such as DBADM. Examples of the usage of administrative authorities are:

```
GRANT SYSOPR TO USER1
GRANT DBADM ON DATABASE RACFDB2 TO USER1
```

There are three installation administrative authorities in DB2 10: SECADM, SYSADM, and SYSOPR. One or two IDs can be assigned the installation SYSADM authority, the SECADM authority, or the installation SYSOPR authority. These are similar to SYSADM, SECADM, and SYSOPR, respectively. However, DB2 does not record these authorities in the catalog, but they are defined in the subsystem initialization parameter module DSNZPARM. No other ID can revoke these installation authorities. These authorities are also allowed to perform some special actions such as running the CATMAINT utility, accessing DB2 when the subsystem is started with ACCESS(MAINT), or starting the directory and catalog databases when they are stopped or in restricted status and running all allowable utilities against these databases. For further details about the installation SYSADM and installation SYSOPR administrative authorities, see Chapter 5, “Managing access through authorization IDs and roles,” in *DB2 10 for z/OS Administration Guide*, SC19-2968.

Implicit privileges are related to the ownership of an object. Ownership is set at object creation time. When the user is the owner of the DB2 object, the user implicitly holds certain privileges over that object. The implicit privileges of ownership are different for each different object. As an example, for a table, these are the implicit privileges:

- Alter or drop the table or any indexes on it.
- Lock the table, comment on it, or label it.
- Create an index or view for the table.
- Select or update any row or column.
- Insert or delete any row.
- Use the LOAD utility for the table.
- Define referential constraints on any table or set of columns.
- Create a trigger on the table.

In order for a user to have implicit authority over an object, the object owner needs to be either that user's primary authorization ID, one of the user's secondary authorization IDs, or the name of a role associated with the user in a trusted context.

Note: When an object is created within a trusted context by an authorization ID with a role associated, and if this role is defined with ROLE AS OBJECT OWNER, the role is the owner of the object no matter whether the object is qualified or not.

The DB2 default access is none. Until access is granted, nothing can be accessed. The DB2 models are:

- Direct: Check the user for authority to the data.
- Indirect: Check at bind time for the application process to the data, and at run time, check the user to the application process.

Cascading revoke

When access is revoked from a certain DB2 authorization ID, all access granted by that authorization ID to the same resource is revoked as well. Assume the following scenario:

- User U1 grants a privilege P to user U2 on an object.
- User U2 grants the same privilege P to user U3 on the same object.

In DB2 9, when U1 now revokes the privilege P from user U2, user U3 also loses its privilege on that object. With DB2 10, you can REVOKE without cascade, that is, stating NOT INCLUDING DEPENDENT PRIVILEGES. Refer to 3.8, "Revoking without cascade" on page 48 for more information.

Time sequence

If the same privilege is granted to a user from two different authorization IDs and if that user in its turn grants that privilege to yet another user, the time in which these grants are given is important to determine what happens when one of the first users revokes the privilege. Consider the following scenario:

- User U1 grants privilege P to user U3 at time T1.
- User U2 grants privilege P to user U3 at time T2.
- User U3 grants privilege P to user U4 at time T3.

When user U1 now revokes the privilege P from user U3 with cascade, the chronological order of T2 and T3 is important to determine if user U3 will keep its privilege or not:

- If T3 comes after T2, user U4 does not lose the privilege because at T3, user U3 might have granted the privilege on the basis of the privilege that was granted by U2.
- If T3 comes before T2, user U4 loses its privilege because user U3 might have granted the privilege only on the basis of the privilege that was granted by user U1.

9.3 RACF managed security

As described earlier, even when only the internal DB2 mechanism is used to secure DB2 resources, RACF might already have been used to:

- Control connections to the DB2 subsystem.

  The ability of a user or address space to connect to DB2 is controlled through checks in the DSNR RACF resource class.
- Assign identities:
  - The DB2 primary authorization ID is a RACF identity. Secondary authorization IDs are often derived by a sign-on exit routine from the RACF-generated list of groups.
- Row level security

  RACF is used with native DB2 security when multilevel security (MLS) and SECLABELs are used with DB2 V8 and later.
- Protect the underlaying DB2 data store.

  The underlaying data sets of DB2 can be protected by RACF.

Important: For any security implementation, either with internal DB2 security or RACF, the underlying DB2 data sets *must* be protected by RACF data set access controls. The list of data sets requiring protection should include:

- VSAM linear data sets for catalog and user objects
- DB2 recovery log and archive data sets
- DB2 BSDS
- Work files used by utilities, bind, and precompiler
- Image copies
- IBM FlashCopy® volumes and data sets
- DB2 libraries SDSNLOAD and SDSNEXIT, which are APF authorized

In addition to DB2 internal security, you can use RACF to control access to DB2 objects, authorities, commands, and utilities by making use of the RACF access control module. This RACF access control module is activated at the DB2 access control authorization exit point DSNX@XAC by replacing the default DB2-provided exit routine by a version that allows RACF checking.

9.3.1 The RACF access control module

The RACF access control module provides a mechanism to:

- Control and audit resources for multiple DB2 subsystems from a single point
- Define security rules before a DB2 object is created
- Preserve security rules for dropped DB2 objects

- Protect multiple DB2 objects with a single security rule using a combination of RACF generic, grouping, and member profiles
- Validate a user ID before giving it access to a DB2 object
- Preserve DB2 privileges and administrative authorities
- Provide flexibility for multiple DB2 subsystems with a single set of RACF profiles
- Administer DB2 security with a minimum of DB2 skills
- Eliminate the DB2 cascading revoke

The RACF access control module is invoked:

- Once at DB2 subsystem startup to perform any required setup prior to authorization checking. Authorization profiles are loaded during this invocation.
- For each DB2 authorization request. This corresponds to the point when DB2 accesses security tables in the catalog to check authorization on privileges.
- Once at DB2 subsystem termination to perform its cleanup before DB2 stops.

DB2 provides a default exit, DSNX@XAC, that is available as a load module in the SDSNLOAD data set. Its source is in SDSNSAMP in member DSNXSXAC. This version of the exit routine returns a code to its invoker, indicating that a user-defined access control authorization exit is not available.

A sample replacement exit source is available in data set SDSNSAMP in member DSNXRXAC. This source can be compiled to replace the dummy-provided exit. Installation job DSNTIJEX provides a step to assemble and link-edit the routine and to place it in the APF-authorized library SDSNEXIT. The load module name or alias name of the access control authorization exit must be DSNX@XAC. The exit routine must have a CSECT name and an entry point with the same name DSNX@XAC.

Note: Starting in DB2 8, the RACF access control module DSNX@XAC is provided by DB2. In earlier *versions*, the exit was provided by RACF.

DB2 provides two sources in SDSNSAMP:

- DSNXSXAC: Dummy default version. A compiled version is present in SDSNLOAD for use when not using external security.
- DSNXRXAC: Sample version that allows the use of RACF for external control of DB2 resources. An assemble and link-edit is required to replace the default version.

9.3.2 RACF profiles and class structure

RACF stores all information about users and resources in profiles. A profile is a record containing RACF information that has been defined by the security administrator.

Each RACF-defined user has a user profile containing information about his or her identity, user attributes, group, and password. Using information in its profiles, RACF provides authentication and access control services. In addition to defining individual user profiles in RACF, you can define group profiles. A group profile defines a group of users. Users who are members of groups can share common access authorities. The use of RACF groups are a common way to implement DB2 based access control through the use of connected groups and granting privileges to a group name, rather than individual RACF authorization IDs.

A class is a collection of resources with similar characteristics. Resources are assigned to various classes. You can control the access to a resource in the class by defining profiles in the class. The authority to access a resource is kept in an access list that is part of the resource profile. The authority can be granted to a user or to a group. RACF supplies a class descriptor table (CDT) that contains a list of default resource class names. You can also define your own class names and use these classes instead of or next to the default provided names.

There are currently 15 DB2 object types that can be defined to RACF as a class. When using RACF for resource security, RACF provides a member class and a grouping class for each DB2 object. Member classes represent single objects or multiple similar named objects if generics are used. A member class name starts with the character M. For a complete list of RACF supplied resource classes, refer to Chapter 12, "Supplied RACF resources classes for DB2", in *DB2 10 RACF Access Control Module Guide*, SC19-2892.

An example is *MDSNDTB*, which is the DB2 class name for a table object, where:

- M: Member class
- DSND: &CLASSNMT value, set in the sample exit, which defaults to DSND
- TB: Table

Grouping classes can represent one or more objects that have similar security requirements. You can first define a group profile in the grouping class and then add individual members to that profile. A grouping class name starts with the character G. An example is *GDSNTB*, where:

- G: Grouping class
- DSND: &CLASSNMT value, set in the sample exit, which defaults to DSND
- TB: Table

The DB2 RACF access control module can be customized to choose a classification model. The format of the class names for DB2 objects depends on the classification model chosen. Classification model 1 uses one set of classes for each different DB2 subsystem, and classification model 2 uses one set of classes for all DB2 subsystems. Using the sample exit version provided by DB2 allows the RACF access control module to use the default DB2 resource classes as defined in the RACF-supplied class descriptor table (CDT).

The customizing of the assembly of DSNXRXAC is obtained through specifying different values for the SET symbols at the beginning of the assembler source. Table 9-1 lists the options that define the class and resource profile names.

Table 9-1 RACF/DB2 external security module SET options

SET symbol	Default value	Description
&CLASSOPT	2	Specifies the classification option to be used: 1 = Single subsystem scope 2 = Multiple subsystem scope
&CLASSNMT	DSN	Specifies the DB2 system name format to be used.
&CHAROPT	1	Specifies the one character name suffix to be used.

&CLASSOPT allows you to choose a single or multiple subsystem scope.

Characteristics of a multiple subsystem scope (classification model 2) include:

- One set of general resource classes that protect multiple subsystems.
- General resource names are prefixed with the DB2 subsystem name.
- Classes provided in the IBM-supplied CDT are multisystem scope.

- You can protect multiple subsystems with a single set of resources.
- Fewer classes overall.

Characteristics of a single subsystem (classification model 1) scope include:

- One set of general resource classes dedicated to one subsystem.
- General resource names are not prefixed with a DB2 subsystem name.
- Classes must be defined by the installation.
- Segregates resources by subsystem.
- There are fewer profiles per class.

For example, if you are running a DB2 subsystem with the name DB0B in a multiple subsystem scope, when a table is accessed, the RACF access control module generates a resource access check of the form of DB0B.*table-owner.table-name.privilege* in the class MDSNTB, which has the format of abbbbccd, where:

a	M for member class or G for grouping class
bbbb	&CLASSNMT with default = DSN
cc	Type of object = TB
d	&CHAROPT but ignored if &CLASSNMT = DSN

Running in a single subsystem scope environment, the resource name is *table-owner.table-name.privilege* with a class name of MDB0BTB1, and is installation defined. Again, the format is abbbbccd, where:

a	M for member class or G for grouping class
bbbb	&CLASSNMT = DB0B
cc	Type of object = TB
d	&CHAROPT with default = 1

In a single subsystem scope environment, to protect the same *table-owner.table-name.privilege* in multiple DB2 subsystems would require multiple class names to be defined. The class names used in the DB2 access control module generated with the default values correspond to the default RACF-provided class names.

9.3.3 RACF defined administration privilege profiles

In addition to classes used to protect specific DB2 privileges for DB2 objects, there is one additional class for the DB2 administrative authorities. The generated class name has the format of *yyyyADMz*, where:

yyyy	&CLASSNMT with default = DSN
ADM	Fixed character string designated for administrative authority classes
z	&CHAROPT with default =1 but ignored if &CLASSNMT = DSN

The class name used in the sample DB2 access control module without changing any of the defaults is DSNADM. In the previous example for the single scope system, the administrative authority class name would be DB0BADM1.

For further details about customizing the RACF access control module, refer to Chapter 3, “Installing the RACF access control module”, in *DB2 10 for z/OS RACF Access Control Module Guide*, SC19-2982.

9.3.4 Activating and using RACF classes

RACF classes must be activated through a RACF SETROPTS command. When additional classes have been set up and activated, DB2 must be restarted.

After the DB2 resource classes are active, RACF profiles are defined in these classes. The way DB2 builds the profile format is to use a combination of the resource name and the name of the privilege required to access that resource.

For further details about the use of the RACF DB2 profiles, refer to Chapter 7, “Making your new RACF resources effective”, in *DB2 10 for z/OS RACF Access Control Module Guide*, SC19-2982.

The order in which the RACF access control module checks user authorization is to first perform some special checking before any RACF checking is done. If applicable, the first test is to check if a user owns the resource or if the user’s name matches the schema name. Then, RACF checks against specific DB2 privileges. If the check to access a specific resource does not allow access, additional DB2 privilege checks on administrative privileges are performed. Administrative authorities have the same meaning as in the DB2 internal security environment.

As an example, to SELECT from a table, an authorization ID must at least have one of the following:

- Ownership of the table
- SELECT privilege on the table
- DBADM authority for the database
- DATAACCESS
- DBADM on SYSTEM
- SYSCTRL (catalog tables only)
- SYSADM authority

When a user is the owner of a DB2 object, that user might have some implicit privileges, but not all privileges associated with the object. Table 9-2 shows how the RACF access control module supports specific DB2 objects with their associated implicit privileges of ownership.

Table 9-2 DB2 objects and associated implicit ownership privileges

DB2 object	**Implicit privileges**
Java archive (JAR)	USAGE
Package	BIND COMMENT ON COPY
Plan	BIND COMMENT ON
Schema	ALTERIN COMMENT ON DROPIN
Sequence	ALTER COMMENT ON USAGE

DB2 object	Implicit privileges
Stored procedure	DISPLAY EXECUTE START STOP

Table	All privileges except:
	CREATE SYNONYM
	DROP SYNONYM
	CREATE VIEW

User-defined distinct type	USAGE
User-defined function	DISPLAY EXECUTE START STOP
View	COMMENT ON DROP

Table 9-3 gives an overview of the DB2 administrative authority profiles in the RACF class DSNADM and MDSNSM. The RACF access control module is bypassed for installation SYSADM and installation SYSOPR.

Table 9-3 Overview of RACF profiles in DSNADM and MDSNSM classes

RACF class	DB2 object	DB2 authority	RACF profile
DSNADM	System	SYSADM	DB2-subsystem.SYSADM
		SYSCTRL	DB2-subsystem.SYSCTRL
		SYSOPR	DB2-subsystem.SYSOPR
		SECADM	DB2-subsystem.SECADM
		DBADM	DB2-subsystem.DBADM
		ACCESSCTRL	DB2-subsystem.ACCESSCTRL
		DATAACCESS	DB2-subsystem.DATAACCESS
	Database	DBADM	DB2-subsystem.database-name.DBADM
		DBCTRL	DB2-subsystem.database-name.DBCTRL
		DBMAINT	DB2-subsystem.database-name.DBMAINT
	All collections	PACKADM	DB2-subsystem.PACKADM
	Specific collection	PACKADM	DB2-subsystem.collection-ID.PACKADM

RACF class	DB2 object	DB2 authority	RACF profile
MDSNSM or GDSNSM	System	SQLADM	DB2-subsystem.SQLADM
		SYSDBADM	DB2-subsystem.SYSDBADM
		BINDADD	DB2-subsystem.BINDADD
		BINDAGENT	DB2-subsystem.owner.BINDAGENT
		CREATEALIAS	DB2-subsystem.CREATEALIAS
		CREATEDBA	DB2-subsystem.CREATEDBA
		CREATESG	DB2-subsystem.CREATESG
		CREATESG	DB2-subsystem.CREATETMTAB
		CREATESECURE OBJECT	DB2-subsystem.CREATESECUREOBJECT
		DEBUGSESSION	DB2-subsystem.DEBUGSESSION
		DISPLAY	DB2-subsystem.DISPLAY
		EXPLAIN	DB2-subsystem.EXPLAIN
		MONITOR1/2	DB2-subsystem.MONITOR1/2
		BSDS	DB2-subsystem.BSDS
		RECOVER	DB2-subsystem.RECOVER
		ARCHIVE	DB2-subsystem.ARCHIVE
		STOPALL	DB2-subsystem.STOPALL
		STOSPACE	DB2-subsystem.STOSPACE
		TRACE	DB2-subsystem.TRACE

RACF uses the terminology *resource* to identify what is being checked by the resource manager. RACF finds the best fitting or most specific profile that covers that resource name and uses the contents of the profile to determine if access should be allowed.

A profile can represent a specific object in a class, but can also be generic to represent a group of objects.

An example of a discrete profile for a specific table is DB0B.DSN8810.EMP.SELECT. In this example, a check is made for READ authority to the resource 'DB0B.DSN88.EMP.SELECT' in the MDSNTB class.

A generic profile for the same table that covers all table privileges is DB0B.DSN8810.EMP.*. The replacement of the specific privilege by an asterisk (*) makes the profile generic, and thus it represents all privileges for the EMP table accessed in the DB2 subsystem DB0B.

Note: Be sure that SETR GENERIC(MDSNTB) has been specified before you create a profile in the MDSNTB class or the profile will be discrete, not generic.

A profile contains the universal access for the object for which it is defined, that is, the access that any user needs to have to access to this resource. The RACF RDEFINE command creates profiles. The RACF command to create a discrete profile in the MDSNTB class so that nobody is to delete the EMP table is:

```
RDEF MDSNTB 'DB0B.DSN8810.EMP.DELETE' UACC(NONE)
```

The profile that is equivalent to granting all table privileges to PUBLIC for the EMP table on subsystem DB0B is:

```
RDEF MDSNTB 'DB0B.DSN8810.EMP.*' UACC(READ)
```

For PUBLIC AT ALL LOCATIONS, the command is:

```
RDEF MDSNTB '*.DSN8810.EMP.*' UACC(READ)
```

The next step is to add an individual user or groups of users to the access list of these resource profiles, thus giving them individual access to these profiles. To perform this task, use the RACF PERMIT command. The RACF commands to allow user DB2R5 to be the only user to delete from the EMP table are:

```
RDEF MDSNTB 'DB0B.DSN8810.EMP.DELETE' UACC(NONE)
PE 'DB0B.DSN8810.EMP.DELETE' CLASS(MDSNTB) ID(DB2R5) ACC(READ).
```

Presuming that we are working on an active DB2 system where the MDSNTB class is activated, after a profile has been added or modified, you must refresh the RACF profiles with the RACF SETROPTS REFRESH command, such as:

```
SETROPTS RACLIST(MDSNTB) REFRESH.
```

The DB2 access control module checks the corresponding RACF resource class profiles (RACF checks for READ authority to the corresponding resource name) and attempts to make an authority decision. If the DB2 access control module cannot make a pass or fails decision, it defers the decision to native DB2 authority checking. This allows us to implement DB2 RACF security one object at a time. For more details about the actual decision tree the DB2 access control module makes, refer to *DB2 UDB for z/OS RACF Access Control Module Guide*, SC18-7433.

9.3.5 RACF sample scenario

As an example of how to use RACF external security, we first re-assembled the DB2-provided RACF access control module with the source that is provided in data set SDSNSAMP in member DSNXRXAC. We copied member DSNTIJEX from SDSNSAMP to our private data set and modified it to only execute the assembly of DSNXRXAC into SDSNEXIT.

With this new version of the RACF access control module, we stopped and started DB2. The RACF access control module generated the output shown in Example 9-1 in the MSTR address space log.

Example 9-1 DB2 MSTR authorization exit routine failure

```
DSNX210I  -DB0B DSNXACAE - ACCESS CONTROL  842
AUTHORIZATION EXIT ROUTINE (DSNX@XAC) HAS INDICATED THAT IT SHOULD
NOT BE CALLED, HAS ABENDED OR HAS RETURNED AN INVALID RETURN CODE
DURING INITIALIZATION. RETURN CODE=000C, REASON CODE=00000004.
CUMULATIVE ABENDS DURING EXIT PROCESSING=0000.
EXIT ROUTINE STATUS: STOPPED.
```

There is insufficient information contained in this message to completely diagnose the problem with the initialization of the RACF authorization routine. To see the complete picture, additional diagnostic information is placed in the z/OS SYSLOG, as shown in Example 9-2.

Example 9-2 RACF access control module output with inactive RACF classes

```
*IRR901A  RACF/DB2 EXTERNAL SECURITY MODULE FAILED TO INITIALIZE
          FOR DB2 SUBSYSTEM DB0B BECAUSE NO ACTIVE DB2 RELATED CLASSES
          WERE FOUND.
*IRR912I NATIVE DB2 AUTHORIZATION IS USED.
 IRR908I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B HAS
         A MODULE VERSION OF PM28543 AND A MODULE LENGTH OF 00007838.
 IRR909I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B
         IS USING OPTIONS: &CLASSOPT=2
                           &CLASSNMT=DSN
                           &CHAROPT=1
                           &ERROROPT=1
                           &PCELLCT=50
                           &SCELLCT=50
 IRR910I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B
         INITIATED RACLIST FOR CLASSES:
          MDSNDB   MDSNPK   MDSNPN   MDSNBP   MDSNCL
          MDSNTS   MDSNSG   MDSNTB   MDSNSM   MDSNSC
          MDSNUT   MDSNUF   MDSNSP   MDSNJR   MDSNSQ
          DSNADM
 IRR911I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B
         SUCCESSFULLY RACLISTED CLASSES:
          * NONE *

 DSNX210I  -DB0B DSNXACAE - ACCESS CONTROL
 AUTHORIZATION EXIT ROUTINE (DSNX@XAC) HAS INDICATED THAT IT SHOULD
 NOT BE CALLED, HAS ABENDED OR HAS RETURNED AN INVALID RETURN CODE
 DURING INITIALIZATION. RETURN CODE=000C, REASON CODE=00000004.
 CUMULATIVE ABENDS DURING EXIT PROCESSING=0000.
 EXIT ROUTINE STATUS: STOPPED.
```

Message IRR901A tells us that initialization failed because the DB2 classes are not active. IRR912I indicates that DB2 has reverted to using native DB2 authorization. Message IRR909I gives the options chosen with the preparation of the sample security exit, and the exit is using classification model 2. Message IRR910I lists the defined classes known to RACF, and IRR911I indicates that none of these classes have been activated, through RACLST.

For our data access scenario, user DB2R1 is able to read data from the EMP table by virtue of being SYSADM, while DB2R5, who has no administrative privileges, is not allowed access. This scenario is illustrated in Example 9-3.

Example 9-3 SQL authorization error

```
SELECT * FROM DSN81010.EMP;
--+---------+---------+---------+---------+---------+---------+-------
DSNT408I SQLCODE = -551, ERROR:  DB2R5 DOES NOT HAVE THE PRIVILEGE TO PERFORM
         OPERATION SELECT ON OBJECT DSN81010.EMP
DSNT418I SQLSTATE   = 42501 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSNXOSC SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = -10 0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
```

```
DSNT416I SQLERRD    = X'FFFFFFF6'  X'00000000'  X'00000000'  X'FFFFFFFF'
X'00000000'  X'00000000' SQL DIAGNOSTIC INFORMATION
```

Based on the IRR910I message shown in Example 9-2 on page 192, the next step is to activate RACF resource classes for DB2. Example 9-4 shows the commands we use for this action.

Example 9-4 RACF commands to activate DB2 resource classes

```
//RUNDB2   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//SYSTSIN  DD   *
 SETROPTS CLASSACT(DSNADM) GENERIC(DSNADM)
 SETROPTS CLASSACT(MDSNPK) GENERIC(MDSNPK)
 SETROPTS CLASSACT(MDSNUT) GENERIC(MDSNUT)
 SETROPTS CLASSACT(MDSNSQ) GENERIC(MDSNSQ)
 SETROPTS CLASSACT(DSNADM) GENERIC(DSNADM)
 SETROPTS CLASSACT(MDSNBP) GENERIC(MDSNBP)
 SETROPTS CLASSACT(MDSNCL) GENERIC(MDSNCL)
 SETROPTS CLASSACT(MDSNDB) GENERIC(MDSNDB)
 SETROPTS CLASSACT(MDSNJR) GENERIC(MDSNJR)
 SETROPTS CLASSACT(MDSNPN) GENERIC(MDSNPN)
 SETROPTS CLASSACT(MDSNSC) GENERIC(MDSNSC)
 SETROPTS CLASSACT(MDSNSG) GENERIC(MDSNSG)
 SETROPTS CLASSACT(MDSNSM) GENERIC(MDSNSM)
 SETROPTS CLASSACT(MDSNSP) GENERIC(MDSNSP)
 SETROPTS CLASSACT(MDSNTB) GENERIC(MDSNTB)
 SETROPTS CLASSACT(MDSNTS) GENERIC(MDSNTS)
 SETROPTS CLASSACT(MDSNUF) GENERIC(MDSNUF)
/*
```

After activating the DB2 resource classes, we again stop and start DB2. This time we receive the output from the RACF access control module, seen in the z/OS SYSLOG shown in Example 9-5.

Example 9-5 RACF access control module output with DB2 resource classes active

```
IRR908I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B HAS
        A MODULE VERSION OF PM28543 AND A MODULE LENGTH OF 000078E8.
 IRR909I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B
        IS USING OPTIONS: &CLASSOPT=2
                          &CLASSNMT=DSN
                          &CHAROPT=1
                          &ERROROPT=1
                          &PCELLCT=50
                          &SCELLCT=50
 IRR910I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B
        INITIATED RACLIST FOR CLASSES:
         MDSNDB   MDSNPK   MDSNPN   MDSNBP   MDSNCL
         MDSNTS   MDSNSG   MDSNTB   MDSNSM   MDSNSC
         MDSNUT   MDSNUF   MDSNSP   MDSNJR   MDSNSQ
         DSNADM
 IRR911I RACF/DB2 EXTERNAL SECURITY MODULE FOR DB2 SUBSYSTEM DB0B
        SUCCESSFULLY RACLISTED CLASSES:
         MDSNDB   MDSNPK   MDSNPN   MDSNBP   MDSNCL
         MDSNTS   MDSNSG   MDSNTB   MDSNSM   MDSNSC
         MDSNUT   MDSNUF   MDSNSP   MDSNJR   MDSNSQ
         DSNADM
```

Again, we try to read the EMP table with both user IDs, and again discover that DB2R1 is able to read from the EMP table, while DB2R5 is not. If access is denied from RACF, we would see RACF message ICH408I for user ID DB2R5. Because this was not the case, this proves that authority checking is still being deferred to DB2-managed authorization checking, and that the access was denied by managed DB2 security. This is because the DB2 access control module detected no object profile in the active MDSNTB class covering the read action, and no administrative profile in the active DSNADM class allowed access.

To make sure that resource control was done by RACF over all resources, we provide a “default” generic profile that protects all of the resources in the MDSNTB class by giving it a universal access of NONE. We achieve this action by submitting the job shown in Example 9-6.

Example 9-6 Defining a “default” generic profile to protect all classes

```
//RUNDB2   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//SYSTSIN  DD   *
 RDEF MDSNTB ** UACC(NONE)
 SETROPTS RACLIST(MDSNTB) REFRESH
/*
```

Note that after these commands we did not stop and restart the DB2 system. The SETR RACLIST REFRESH updates the RACF security information.

Neither of the two user IDs is able to read from the table. Example 9-7 shows that access is denied by RACF.

Example 9-7 Access denied by RACF for DB2R1 and DB2R5

```
ICH408I USER(DB2R1 ) GROUP(SYS1    ) NAME(MLS TEST USER       )
  DB0B.DSN8810.EMP.SELECT CL(MDSNTB  )
  INSUFFICIENT ACCESS AUTHORITY
  FROM ** (G)
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
ICH408I USER(DB2R5 ) GROUP(SYS1    ) NAME(MLS TEST USER       )
  DB0B.DSN8810.EMP.SELECT CL(MDSNTB  )
  INSUFFICIENT ACCESS AUTHORITY
  FROM ** (G)
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

To allow DB2R5 to select again from the EMP table, we now add to the MDSNTB class a discrete profile with the SELECT privilege for EMP that has universal access NONE. Also, instead of directly granting READ to DB2R5, instead we permit READ to a group, GRPA, which is connected through RACF group list processing to DBB2R5.

Example 9-8 shows a series of RACF commands that create a group, connect the RACF ID DB2R5 to the group GRPA, and list the IDs connected to GRPA.

Example 9-8 RACF commands to create and connect a group to a RACF ID

```
ADDGROUP GRPA
CONNECT DB2R5 GROUP(GRPA)
LG GRPA
```

We then add a profile with READ access for GRPA to the access list of this newly created discrete profile. We do this by submitting the job shown in Example 9-9.

Example 9-9 Giving DB2R5 SELECT access on table EMP through group GRPA

```
//RUNDB2   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//SYSTSIN  DD   *
 RDEF MDSNTB DB0B.DSN8810.EMP.SELECT UACC(NONE)
 PERMIT DB0B.DSN8810.EMP.SELECT CLASS(MDSNTB) ID(GRPA) ACC(READ)
 SETROPTS RACLIST(MDSNTB) REFRESH
/*
```

To give SELECT access to all resources owned by DSN8810 on system DB0B, we could have defined a generic profile DB0B.DSN8810.*.SELECT with universal access READ. However, we prefer to give access to DB2R5 to only a selected number of tables owned by DSN8810.

RACF can also be used to protect system administrative privileges. We first issue a DISPLAY THREAD(*) command with user ID DB2R5. Example 9-10 shows the resulting output.

Example 9-10 DB2R5 is denied from DISPLAY THREAD on DB0B

```
*** DSN9016I  -DB0B '-DIS THREAD' COMMAND REJECTED, UNAUTHORIZED REQUEST
DSN9023I  -DB0B DSN9SCND '-DIS THREAD' ABNORMAL COMPLETION
***
```

To assign SYSADM authority to user DB2R5 we submit the RACF commands shown in Example 9-11.

Example 9-11 Giving SYSADM authority to group GRPA

```
//RUNDB2   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//SYSTSIN  DD   *
 RDEF DSNADM ** UACC(NONE)
 RDEF DSNADM DB0B.DSNADM UACC(NONE)
 PERMIT DB0B.DSNADM.SYSADM CLASS(DSNADM) ID(GRPA) ACC(READ)
 SETROPTS RACLIST(DSNADM) REFRESH
/*
```

We then retry the DISPLAY THREAD command on DB0B and receive the results shown in Example 9-12.

Example 9-12 Display thread result when connected to DBADM.SYSADM

```
DSNV401I  -DB0B DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -DB0B ACTIVE THREADS -
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
RRSAF    T       10 DB0BADMT_II STC     ?RRSAF   011D     2
RRSAF    T      470 DB0BADMT_DMN STC     ?RRSAF   011D     3
TSO      T  *     3 DB2R5       DB2R5            011F    27
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -DB0B DSNVDT '-DIS THREAD' NORMAL COMPLETION
***
```

RACF access control and cascade revocation

One of the other major benefits of using the RACF access control exit is that the effects of cascade revocation can be eliminated.

Choosing to use the access control authorization exit routine

Using RACF to perform access control is not the best choice for every customer. Basically, the decision depends on which group, security administrator, or database administrator should be responsible for security administration.

Consider the following points before choosing RACF to perform access control:

- If you want the database administrators to manage security, integration with DB2 is important. Using RACF access control provides less integration with DB2. In most of these cases, DB2 authorization provides advantages.
- If you want security administrators to manage security, integration with the security server is more important. In most of these cases, using RACF for access control provides advantages. Furthermore, if you want a security group to define authorization and a centralized security control point, RACF access control is an excellent match. When RACF is used for access control of DB2 objects, the same SMF records (SMF type 80) records which are used for other resource accesses are used for the DB2 access log.
- If you change from DB2 authorization to RACF access control, you must change to RACF methods for some authorization techniques, and you must understand how DB2 and RACF work together.

Expect to make the following changes when you implement RACF access control:

- Plan to use RACF facilities (such as groups and patterns) more.
- Plan to use patterns instead of individual item access profiles and permissions.
- Plan to use RACF groups instead of secondary authorization IDs, which are not implemented in RACF. OWNER(secondaryID) generally must be a valid group.
- Find an alternative to BINDAGENT. BINDAGENT is based on secondary authorization IDs, which are not implemented in RACF. BINDAGENT provides a relatively weak security separation. Customers have found alternatives.
- Understand how SET CURRENT SQLID works with RACF. SET CURRENT SQLID can set a qualifier, but does not change authorization.

- Know that authorizations are not dropped when objects are dropped or renamed. Be aware of the relationship between objects and revoked privileges.
- Plans and packages are not invalidated when authorizations are revoked. Views are not dropped when authorizations are revoked.

Some customers base this decision on ease of administration, and service level agreements covering how quickly access privileges can be provided to business units. However, as discussed earlier, adherence to external or internal security initiatives and regulatory requirements are now bringing the separation of roles and responsibilities element into this decision. With the RACF approach, subject to the above limitations, there is a clearly demonstrated separation of security administration roles and responsibilities. With DB2 10, customers can further enforce segregation of security tasks between database and security administrators, either through usage of RACF or not.

9.3.6 RACF/DB2 Conversion Utility

The RACF/DB2 Conversion Utility is a tool provided on an *as is* basis for customers. The tool and supporting documentation can be found at the following address:

http://www.ibm.com/systems/z/os/zos/features/racf/downloads/racfdb2r.html

It converts the contents of the SYSIBM.SYSxxxAUTH tables into commands to generate equivalent RACF profiles. The RACF profiles are used in conjunction with the RACF/DB2 External Security Module provided by the IBM OS/390® Security Server product to replace the default DB2 access control authorization exit routine.

The utility requires RXSQL to be installed. RXSQL is an IBM licensed product and is an add-on for REXX/TSO. RXSQL includes a plan that must be bound to the DB2 subsystem. The default name for this plan is RXSQL. If the plan is bound under a different name, a variable must be changed in the RXSADM and RXSRES execs. The variable is RXSQL_DB2PLAN.

Authorization required

To run the RACF/DB2 Conversion Utility, you must have either SYSADM or DATAACCESS or SELECT authority to every SYSIBM.SYSxxxAUTH table.

To execute the generated commands, you must have either RACF Special or CLAUTH to every class used, and all profiles generated must be owned by either:

- The user executing the generated CLIST
- Groups to which the executing user has group-SPECIAL

For ease of use, SYSADM and RACF SPECIAL should be used.

RDB2 exec

RDB2 is the REXX exec that generates commands to define all DB2 authorities to RACF.

The RACF/DB2 Conversion Utility attempts to closely duplicate the DB2 catalog information in RACF profiles. An exact match is not always possible because RACF does not work the same way as DB2 authorization.

In general, the utility does the following for each resource type:

- Discover all distinct entries for a resource type and generate a RACF RDEFINE command to define the entry.
- Generate a PERMIT...RESET command to ensure the access list is clean.

- Discover if PUBLIC was granted to those resources, and if so, generate a RACF RALTER command to change the UACC to READ.
- Discover all authorization IDs that were granted access without GRANT option, and generate a PERMIT command giving READ access.
- Discover all authorization IDs that were granted access with GRANT option, and generate a PERMIT command giving ALTER access.

Because the majority of profiles generated are discrete general resource profiles, ALTER access is similar to access with the GRANT option. If you want your installation to only allow users with RACF ownership or system attributes to be able to modify the DB2 RACF profiles, replace ¢ ACC(ALTER)¢ with ¢ ACC(READ)¢ before executing the CLIST.

DROP INDEX and ALTER INDEX commands do not use the same resource names to check DBADM authority as other commands. The other resource names include a database name. The RACF/DB2 Conversion Utility could either ignore the non-delimited DBADM resource or add all users with DBADM on any database to the non-delimited DBADM resource. Rather than making a decision for everyone, the RACF/DB2 Conversion Utility creates an optional CLIST. The optional CLIST includes statements to define and permit all the users with DBADM to any database. The customer can choose to execute it or not.

The RACF/DB2 Conversion Utility assumes that all DB2 authorization IDs exist in RACF as either users or groups. If that is not true at your installation, you have a residual access security exposure. In addition, the RACF profiles generated require valid RACF users or groups. Ensure that all authorization IDs are defined to RACF prior to executing the RACF/DB2 Conversion Utility.

Resources added to DB2 after DB2 V5, such as functions or types or roles, are not supported. The utility is a starting point only; it does not take into account the simplification of rules that can be done using generic profiles or grouping profiles. It also create many PERMITs that could be deleted.

The RACF/DB2 Conversion Utility creates two CLISTs allocated by the DDNAMES CLIST and OPTCLST. CLIST contains the RACF commands to match the DB2 catalog security. OPTCLST contains optional RACF commands. In this example, the STEPLIB DD statement points to the RXSQL library, as well as the DB2 load library. The SYSPROC and SYSEXEC DD statements point to the RACF/DB2 Conversion Utility library. The CLIST DD statement defines the data set where the generated commands from RXSADM are written. The RCLST DD statement defines the data set where the generated commands from RXSRES are written. The SYSTSIN DD statement calls the RXSADM exec and defines the parameters. Example 9-13 shows a sample JCL to execute the DB2RACF REXX sample utility.

Example 9-13 JCL example to execute the DB2RACF REXX sample

```
//STEP1  EXEC  PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSPROC  DD  DISP=SHR,DSN=DB2R1.DB2RACF.REXXPDS(RACFDB2)
//CLIST    DD  DISP=(NEW,CATLG),DSN=DB2R1.RACFDB2.CONVCLST,
//   UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=255)
//*PTCLST  DD  DISP=SHR,DSN=RCF.RACFDB2.OPTCLST
//OPTCLST  DD  DISP=(NEW,CATLG),DSN=DB2R1.RACFDB2.OPTCLST,
//   UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=255)
//SYSTSIN  DD  *
  EXECUTIL SEARCHDD(YES)
  %RACFDB2 DB2R1  DBOA DSN      2
//*        OWNER  DBOA CLASSMNT MODEL CHAROPT
```

In Example 9-13 on page 198, the following positional parameters are coded as follows:

- OWNER is the RACF owner of the new profiles.
- DB0A is the DB2 subsystem ID to which the DB2RACF utility connects.
- CLASSMNT should match the value chosen in DSNXRXAC.
- MODEL is the classification model and should match the one chosen in DSNXRXAC.
- CHAROPT in this example is left blank, and should match the value chosen in DSNXRXAC.

Note: The JCL that comes with the RACF/DB2 Conversion Utility uses the same defaults as the RACF/DB2 External Security Module. Any changes to the RACF/DB2 External Security Module must be reflected in the exec parameters.

When executed, the RACF/DB2 Conversion Utility generates two separate sets of RACF commands stored in two separate data sets. Example 9-14 show a sample of generated commands.

Example 9-14 Sample RACF/DB2 conversion utility output

```
RDEF DSNADM DB0A.DB2OSC.DBADM UACC(NONE) AUDIT(ALL(READ)) OWNER(DB2R1)
PE DB0A.DB2OSC.DBADM CLASS(DSNADM) RESET
RDEF DSNADM DB0A.DB2PM.DBADM UACC(NONE) AUDIT(ALL(READ)) OWNER(DB2R1)
PE DB0A.DB2PM.DBADM CLASS(DSNADM) RESET
RDEF DSNADM DB0A.DB2PMAUD.DBADM UACC(NONE) AUDIT(ALL(READ)) OWNER(DB2R1)
PE DB0A.DB2PMAUD.DBADM CLASS(DSNADM) RESET
```

Note: In our test environment, with an extremely small catalog, the resultant data set generated in excess of 23,000 RACF statements. In mature DB2 installations, with large authorization tables in the DB2 catalog, expect to generate large RACF command data sets. It is a best practice to not run these commands as generated, and instead use the output as a guide to build generic groups.

Review the commands generated. The RACF/DB2 Conversion Utility does not use the grouping classes. Evaluate any resources with equivalent access requirements to determine if any profiles should be replaced by grouping profiles or generic profiles.

The RACF/DB2 Conversion Utility does not generate a ssid.DBADM profile that is required for access to DROP and ALTER INDEX privileges through DBADM. Review whether all users with the DBADM privilege to any database or a subset should also be given access to ssid.DBADM.

When the RACF/DB2 Conversion Utility creates commands for Package privileges, it generates generic profiles where the package name is *. The RACF/DB2 Conversion Utility copies the access given to the generic profile to all more specific profiles. The RACF/DB2 Conversion Utility does the same for Collection privileges.

9.3.7 RACF SMF data unload utility IRRADU00

RACF audit data is a record of an installation's security relevant events. This data is used to verify the effectiveness of an installation's security policy, determine whether the installation's security objectives are being met, and identify unexpected security relevant events.

The RACF SMF data unload utility (IRRADU00) enables installations to create a sequential file from the security relevant audit data. The sequential file can be used in several ways:

- Viewed directly
- Used as input for installation-written programs
- Manipulated with sort/merge utilities
- Output to an XML formatted file for viewing on a web browser
- Uploaded to a database manager (for example, DB2) to process complex inquiries and create installation-tailored reports

It is not intended to be used directly as input to RACF commands.

For the DB2 customer who has implemented RACF managed DB2 security, the data obtained from IRRADU00 can be loaded into DB2 tables and then combined with other audit data, for example, output from the execution of the IFI audit trace, processed by Tivoli OMEGAMON XE for DB2 Performance Expert Version V5R1 (OMEGAMON PE), and loaded into the OMEGAMON PE performance database audit tables.

The IRRADU00 utility processes these types of SMF records:

- Type 30 Job initiation - Subtype 1 (Job initiation) and subtype 5 (Job termination)
- Type 80 Resource access - No subtypes in record
- Type 81RACF initialization - No subtypes in record
- Type 83 Data sets affected by a security label change - Subtype 1, EIM - Subtype 2, LDAP - Subtype 3, and Remote audit - Subtype 4

Of particular interest to DB2 are the contents of the SMF TYPE 80 records. The SMF dump utility (IFASMFDP or IFASMFDL) is used to select records based on the date, time, and SMF system identifier. The RACF SMF data unload utility uses the SMF Dump Utilities (IFASMFDP or IFASMFDL) as the *driver* module to control its invocation. The RACF SMF data unload utility is invoked as USER2 and USER3 exits to IFASMFDP or IFASMFDL. To request RACF SMF data unload utility processing, enter the names of the RACF SMF data unload utility modules (IRRADU00 and IRRADU86) in the SYSIN data stream for IFASMFDP.

Example 9-15 shows sample JCL to execute the unload utility.

Example 9-15 IRRADU00 RACF SMF data unload sample JCL

```
//SMFDUMP EXEC PGM=IFASMFDP
//SYSPRINT DD SYSOUT=A
//ADUPRINT DD SYSOUT=A
//OUTDD DD DISP=(,CATLG,DELETE),UNIT=3390,DSN=DB2R1.RACF.IRRADU00,
//  DCB=(RECFM=VB,LRECL=12288,BLKSIZE=23476),
//  SPACE=(CYL,(20,20),RLSE)
//SMFDATA DD DISP=SHR,DSN=SMFDATA.ALLRECS(-1)
//SMFOUT DD DUMMY
//SYSIN DD *
INDD(SMFDATA,OPTIONS(DUMP))
OUTDD(SMFOUT,TYPE(000:255))
ABEND(NORETRY)
USER2(IRRADU00)
USER3(IRRADU86)
/*
```

For the execution of IRRADU00, the following DD statements are required:

- OUTDD defines the single sequential output data set, which contains the various RACF type 80 event records. The output of IRRADU00 is a set of variable length records. This data set must be allocated as a variable length data set, with a logical record length (LRECL) of at least 12288. If a shorter LRECL is supplied, any value less than 12288 is reset to 12288 by the DCB open exit for OUTDD in IRRADU00. IRRADU00 also changes the block size of the data set to be at least four more bytes than the LRECL, unless the block size was set to zero to allow the system to choose the best block size.
- SMFDATA specifies the data set that contains the input SMF data.
- SMFOUT specifies the optional output SMF data. Normally, IFASMFDP is used to copy the system SMF data set, as part of normal system SMF switch and off load processing. IRRADU00 is called as an exit by IFASMFDP, and when control is given back to IFASMFDP, the SMF record is written to the SMFOUT DD. Unless this record needs to be retained for other purposes, this DD statement can be defined as DUMMY.

IRRADU00 execution summary report

When executed, IRRADU00 creates a execution summary report that shows the results of the utility execution, as shown in Example 9-16.

Example 9-16 Sample output execution from IRRADU00

```
IRR67652I The utility processed 867 SMF type 30 records.
IRR67652I The utility processed 1147 SMF type 80 records.
IRR67652I The utility processed 0 SMF type 81 records.
IRR67655I The utility processed 0 SMF type 83 subtype 1 records.
IRR67655I The utility processed 0 SMF type 83 subtype 2 records.
IRR67655I The utility processed 0 SMF type 83 subtype 3 records.
IRR67655I The utility processed 0 SMF type 83 subtype 4 records.
IRR67655I The utility processed 0 SMF type 83 subtype 5 records.
IRR67655I The utility processed 0 SMF type 83 subtype 6 records.
IRR67653I The utility bypassed 35003 SMF records not related to IRRADU00.
IRR67650I SMF data unload utility has successfully completed.
```

The output file from the RACF SMF data unload utility can be:

- Viewed directly
- Used as input to your own programs
- Manipulated with sort/merge utilities
- Used as input to a database management system so you can produce reports tailored to your requirements
- Viewed using a web browser

The records produced by the RACF SMF data unload utility are designed to be processed by the DB2 load utility or its equivalent. The definition and control statements for a DB2 utilization of the output, all of which are contained in SYS1.SAMPLIB, are as follows:

- Sample data definition language (DDL) statements to define the relational presentation of the audit information and sample DB2 definitions that perform database and index creation. They are in member IRRADUTB.
- Sample control statements for the DB2 load utility that map the output from the RACF SMF data unload utility. They are in member IRRADULD.
- Sample structured query language (SQL) queries that demonstrate useful inquiries that can be made. They are in member IRRADUQR.

Simple audit reporting scenario using IRRAUD00

To demonstrate how to use the RACF SMF data unload utility and the RACF DB2 database, consider the following scenario:

1. Create RACF GROUP GRPA.
2. Connect GRPA to RACF ID PAOLOR2 and GRPB to PAOLOR5.
3. Define resource profile for table DB2R5.PARTS with the audit attribute on.
4. Permit READ to GRPA.
5. Permit NONE to GRPB.
6. Execute SELECT on DB2R5.PARTS from both PAOLOR2 and PAOLOR5.

Query the ADDGROUP table to see the audit event for this administration activity. Example 9-17 shows a sample SQL and query result.

Example 9-17 SQL query to show RACF ADDGROUP event

```
SELECT
 AG_EVENT_TYPE,
 AG_EVENT_QUAL,
 AG_TIME_WRITTEN,
 AG_EVT_USER_ID,
 AG_GRP_ID,
 AG_SPECIFIED
 FROM DB2R1.ADDGROUP
 WHERE AG_EVT_USER_ID = 'DB2R1';
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--
AG_EVENT_TYPE  AG_EVENT_QUAL  AG_TIME_WRITTEN  AG_EVT_USER_ID  AG_GRP_ID  AG_SPECIFIED
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--
ADDGROUP       SUCCESS        14.09.49         DB2R1           GRPB       SUPGROUP(SYS1) OWNER(DB2R1) TERMUACC
```

Access the CONNECT table to show the previously created two RACF groups being connected to PAOLOR2 and PAOLOR5. Example 9-18 shows this query and the query results.

Example 9-18 Query of CONNECT showing group connection to GRPA

```
SELECT
CON_EVENT_TYPE,
CON_USER_ID,
CON_SPECIFIED
 FROM DB2R1.CONNECT
 WHERE CON_EVT_USER_ID = 'DB2R1';
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---
CON_EVENT_TYPE  CON_USER_ID  CON_SPECIFIED
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---
CONNECT         PAOLOR2      UACC(NONE) AUTHORITY(USE) NOADSP RESUME NOGRPACC NOOPERATIONS NOAUDITOR NOSPECIAL GROUP(GRPA)
CONNECT         PAOLOR5      UACC(NONE) AUTHORITY(USE) NOADSP RESUME NOGRPACC NOOPERATIONS NOAUDITOR NOSPECIAL GROUP(GRPB)
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
```

Access the DEFINE table to show the definition of the profile protecting the table DB2R5.PARTS, as shown in Example 9-19.

Example 9-19 Query of the DEFINE table showing the profile definition

```
SELECT
 DEF_EVENT_TYPE,
 DEF_EVENT_QUAL,
 DEF_EVT_USER_ID,
 DEF_RES_NAME,
 DEF_CLASS
 FROM DB2R1.DEFINE
 WHERE DEF_CLASS = 'MDSNTB';
---------+---------+---------+---------+---------+---------+---------+--------
```

```
DEF_EVENT_TYPE  DEF_EVENT_QUAL  DEF_EVT_USER_ID  DEF_RES_NAME
---------+---------+---------+---------+---------+---------+---------+--------
DEFINE          SUCCESS         DB2R1            DB0B.DB2R5.PARTS.*
DEFINE          SUCCESS         DB2R1            DB0B.DB2R5.PARTS.SELECT
```

Display the PERMIT table where you permit access to the profile defined previously to the connected groups, as shown in Example 9-20.

Example 9-20   PERMIT table query showing permission to connected GRPA and GRPB

```
SELECT
 PERM_EVENT_TYPE,
 PERM_EVENT_QUAL,
 PERM_SPECIFIED,
 PERM_RES_NAME
 FROM DB2R1.PERMIT;
---------+---------+---------+---------+---------+---------+---------+---------+---------+------
PERM_EVENT_TYPE  PERM_EVENT_QUAL  PERM_SPECIFIED                      PERM_RES_NAME
---------+---------+---------+---------+---------+---------+---------+---------+---------+------
PERMIT           SUCCESS          CLASS(MDSNTB) ID(GRPB) ACCESS(READ)  DB0B.DB2R5.PARTS.*
PERMIT           SUCCESS          CLASS(MDSNTB) ID(GRPA) ACCESS(READ)  DB0B.DB2R5.PARTS.SELECT
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
```

Display the access events generated from SQL select activity against the table protected previously. The first example looks for authorized SQL accesses in the event details stored in the ACCESS table. Example 9-21 show this query.

Example 9-21   ACCESS event showing successful access to a protected resource

```
SELECT
 ACC_EVENT_TYPE,
 ACC_EVENT_QUAL,
 ACC_EVT_USER_ID,
 ACC_CLASS,
 ACC_RES_NAME,
 ACC_REQUEST,
 ACC_GRANT
 FROM DB2R1.ACCESS
 WHERE ACC_EVENT_QUAL = 'SUCCESS' AND ACC_CLASS = 'MDSNTB';
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---
ACC_EVENT_TYPE  ACC_EVENT_QUAL  ACC_EVT_USER_ID  ACC_CLASS  ACC_RES_NAME               ACC_REQUEST  ACC_GRANT
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---
ACCESS          SUCCESS         PAOLOR2          MDSNTB     DB0B.DB2R5.PARTS.SELECT     READ         READ
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
```

Query the ACCESS table looking for unsuccessful access events. Example 9-22 shows the SQL to generate this event.

Example 9-22   SQL to look for unsuccessful attempts to access protected resources

```
SELECT
 ACC_EVENT_TYPE,
 ACC_EVENT_QUAL,
 ACC_EVT_USER_ID,
 ACC_CLASS,
 ACC_RES_NAME,
 ACC_REQUEST,
 ACC_GRANT
 FROM DB2R1.ACCESS
 WHERE ACC_EVENT_QUAL = 'INSAUTH' AND ACC_CLASS = 'MDSNTB';
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--
```

```
ACC_EVENT_TYPE  ACC_EVENT_QUAL  ACC_EVT_USER_ID  ACC_CLASS  ACC_RES_NAME                ACC_REQUEST  ACC_GRANT
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--
ACCESS          INSAUTH         DB2R5            MDSNTB     DBOB.SYSIBM.SYSDUMMY1.SELECT  READ         NONE
ACCESS          INSAUTH         PAOLOR5          MDSNTB     DBOB.SYSIBM.SYSDUMMY1.SELECT  READ         NONE
ACCESS         INSAUTH        PAOLOR5         MDSNTB     DBOB.DB2R5.PARTS.SELECT       READ        NONE
```

It is important to be able to correlate activities that are detected by RACF, such as access violations and changes to permissions and authorities, to events that are detected by DB2, for example, authorization failures detected by DB2. If you store both the RACF detected events, along with DB2 detected events, such as IFI collected audit records into DB2 tables (for example, output from the execution of the IFI audit trace, processed by OMEGAMON Performance Expert, and loaded into the DB2PM[2] performance database audit tables), it is then possible to generate reports using SQL to provide a correlated view of the security environment.

9.3.8 Using the RACF database unload utility IRRDBU00

You should run IRRDBU00 from a copy of the RACF database, which reduces the possibility of impacting performance. There are several different RACF utilities that can be used to create a copy of your RACF database. These include IRRUT200[3] and IRRUT400. We use IRRUT400 to create the example RACF database copy. Details about the use of this utility and IRRUT200 can be found in the *z/OS V1R12.0 Security Server RACF System Programmer's Guide*, SA22-7681. Example 9-23 shows sample JCL used to create our RACF database copy.

Example 9-23 Sample JCL to execute IRRUT400 to create RACF database copy

```
//IRRUT400 EXEC PGM=IRRUT400,PARM='NOLOCKINPUT,FREESPACE(20)'
//SYSPRINT DD SYSOUT=*
//INDD1 DD DISP=SHR,DSN=SYS1.RACFESA
//OUTDD1 DD DSN=DB2R1.SYS1.RACFESA,DISP=(,CATLG),
// SPACE=(CYL,10,,CONTIG),
// VOL=SER=SBOX1T,
// DCB=DSORG=PSU,UNIT=SYSALLDA
```

The output data set should be of sufficient size to contain the RACF database copy. In addition, in our system, there was a single data set RACF database, but depending on the size and configuration of your sysplex, there could be multiple data sets, which would be specified by adding additional INDD DD statements.

Note: You should specify the NOLOCKINPUT parameter to prevent RACF from issuing a hardware RESERVE on the volume where the RACF database is located. With NOLOCKINPUT, be aware that if any RACF administration occurs during the copy operation, these changes might not be reflected in the copy. Activity that updates the RACF database will not be allowed, which includes the RACF administrative commands and other activities that update the database (such as LOGONs). Running with a locked database should be done for short times only, which is why most clients use UT200 for the copy, as it is about as fast as ICEGENER.

[2] DB2PM is the old abbreviation for the tool, and it is still used in the procedures for the OMEGAMON PE performance database.

[3] UT200 tends to be faster, but it has the restriction that it can be used only to copy a data set only to another identically sized data set on the same device type.

After the RACF database copy has been created, it is then used as input for the next step of the process, which is the RACF database unload. The RACF database unload utility enables installations to create a sequential file from a RACF database. The sequential file can be used in several ways:

- Viewed directly
- Used as input for installation-written programs
- Manipulated with sort/merge utilities.

It can also be uploaded to a database manager, such as DB2, to process complex inquiries and create installation-tailored reports.

IRRDBU00 processes either a copy of the RACF database, a backup RACF database, or the active RACF database. You must have UPDATE authority to the database. Run the utility against a recent copy of your RACF database using the NOLOCKINPUT option.

While processing, IRRDBU00 serializes one profile at a time. When IRRDBU00 has finished copying a profile, it releases the serialization.

You can choose for an installation to unload its database with one utility invocation, or if it has split its database, it can unload individual pieces of its database with separate utility invocations. These utility invocations can execute concurrently. For details about further operational considerations when running IRRDBU00, consult the *z/OS V1R12.0 Security Server RACF Security Administrator's Guide*, SA22-7683.

On our system, we have a single data set RACF database, so our IRRDBU00 utility JCL is shown in Example 9-24.

Example 9-24 IRRDBU00 RACF database unload sample JCL

```
//UNLOAD EXEC PGM=IRRDBU00,PARM=NOLOCKINPUT
//SYSPRINT DD SYSOUT=*
//INDD1 DD DISP=SHR,DSN=DB2R1.SYS1.RACFESA
//*NDD2 DD DISP=SHR,DSN=SYS1.RACFDB.PART2.COPY
//*NDD3 DD DISP=SHR,DSN=SYS1.RACFDB.PART3.COPY
//OUTDD DD DISP=(,CATLG,DELETE),DSN=DB2R1.SYS1.RACFDB.FLATFILE,
//   SPACE=(CYL,(20,10),RLSE),UNIT=SYSALLDA,
//   DCB=(RECFM=VB,LRECL=4096,BLKSIZE=0)
```

On our system, we had a small RACF database, so the size of our OUTDD file was small; on a large and mature RACF installation, this file could be significantly larger.

You can use the DB2 load utility or its equivalent to process the records produced by the database unload utility. The definition and control statements for a DB2 utilization of the output, all of which are contained in SYS1.SAMPLIB, are as follows:

- Sample data definition language (DDL) statements to define the relational presentation of the RACF database and sample DB2 definitions that perform database and index creation. They are contained in member RACDBUTB.
- Sample control statements for the DB2 load utility that map the output from the database unload utility (IRRDBU00). They are contained in member RACDBULD.
- Sample structured query language (SQL) queries that perform the following queries. They are contained in member RACDBUQR.
  - Listing all of the members of a group, including a universal group
  - Listing all of the users with the SPECIAL attribute
  - Finding all of the groups to which a user is connected

– Finding all of the data set access lists that contain user IDs that are no longer valid
– Listing of z/OS UNIX user identifiers (UIDs) with associated user ID and programmer name
– Listing of z/OS UNIX group identifiers (GIDs) with associated group name
– Listing of UIDs, including only those users who are connected to a group that has a GID (for each UID, the user ID and programmer name are listed)

To create and manage a DB2 database that contains the output from the database unload utility, you must:

1. Create one or more DB2 databases.
2. Create one or more DB2 table spaces.
3. Create DB2 tables.
4. Create the DB2 indexes.
5. Load data into the tables.
6. Reorganize the data in the tables (optional).
7. Create performance statistics (optional).
8. Delete table data (optional).

The first three steps are initial setup, and you can choose to run them once. When you get new data to import into the DB2 database, you delete your current table data. You then reload and reorganize your tables and create the performance statistics. Details about how to use the sample DDL provided by RACF can be found in *z/OS V1R12.0 Security Server RACF Security Administrator's Guide*, SA22-7683.

On our system, we placed the table space containing the tables that use the data from the RACF database unload into the same database created earlier for the RACF SMF extract utility IRRADU00. We also elected to load all of the extracted data into the appropriate DB2 tables; your requirements and security controls might dictate a different approach. Once loaded, this data can then be queried by SQL. Sample SQL statements can be found in SYS1.SAMPLIB member RACDBUQR. For example, to locate the RACF groups that have been connected to a primary authorization ID, Example 9-25 shows SQL to provide that information.

Example 9-25 SQL query to locate connected RACF groups to a primary RACF ID

```
SELECT
      USCON_NAME
     ,USCON_GRP_SPECIAL
     ,USCON_GRP_ID
     ,USCON_GRP_OPER
     ,USCON_GRP_AUDIT
     ,USCON_REVOKE
     ,USCON_REVOKE_DATE
     ,USCON_RESUME_DATE
FROM
       DB2R1.USER_CONNECT_DATA USCON
WHERE
       USCON_NAME='DB2R1'
ORDER BY
      USCON_GRP_ID
---------+---------+---------+---------+---------+---------+---------+------
USCON_NAME  USCON_GRP_SPECIAL  USCON_GRP_ID  USCON_GRP_OPER  USCON_GRP_AUDIT
---------+---------+---------+---------+---------+---------+---------+------
DB2R1       N                  DB2PM         N               N
```

```
DB2R1       N                 SECADM       N              N
DB2R1       N                 SYS1         N              N
```

You have seen how RACF can be used to authenticate users as they access the DB2 subsystem environment, and how access to DB2 resources and privileges can be controlled by RACF. You have also seen how DB2 can be used to store information about RACF related administration activities and DB2 resource access occurs for SQL based reporting.

We have also demonstrated how authorization and privilege information stored in the DB2 catalog tables can be extracted and formatted into RACF commands. These RACF commands can then be used for migrating from a DB2 controlled security environment to one that is administered from within RACF.

Part 3

Implementation scenarios

In this part, we show details about implementation scenarios.

This part contains the following chapters:

10

Implementing data access control

The DB2 authorization model has been enhanced to allow separation (or segregation) of duties. DB2 10 clearly divides the duties of the database administrator and the security administrator and introduces new authorities that enable you to grant only the access a user needs to do their work.

To take advantage of the granularity of the DB2 administrative authority and simplify your system database administration, you can separate the privileges of the SYSADM authority and migrate them to other administrative authorities based on the security needs of your business. This action minimizes the need for granting the SYSADM authority.

You can separate the SYSADM authority into the SECADM and other administrative authorities by setting the SEPARATE_SECURITY system parameter on panel DSNTIPP1 to YES during installation or migration. With this setting:

- A user who holds the new SECADM authority can now grant and revoke all authorities and privileges, including DBADM and SYSADM authorities.
- A user who holds the SYSADM authority can no longer control access or manage security-related objects (that is, roles, trusted contexts, row permissions, and column masks). The user also cannot grant or revoke privileges that are granted by others.

Refer to Chapter 3, “Administrative authorities and security-related objects” on page 31 for the definition of authorities.

In addition, data access control through row permissions and column masking provides the capability of filtering data depending on the business need of the users. Refer to Chapter 5, “Data access control” on page 87 for information about row permissions and column masking.

In this chapter, we show scenarios with implementation of both of these sets of functions. This chapter contains the following topics:

- Description of the Spiffy Computer Company
- Scenario 1: Separation of duties
- Scenario 2: Classification of users

10.1 Description of the Spiffy Computer Company

The Spiffy Computer Company develops and sells projects. It is composed of the Human Resource department, the Payroll department, and the Project department. Its DB2 application is composed of seven tables, that is, DEPT, EMP, EMP_OTHER_INF, PROJ, ACT, PROJACT and EMPPROJACT.

For more details about the company, refer to Appendix A, “Spiffy Computer Company security setup” on page 389.

10.2 Scenario 1: Separation of duties

A sizeable number of control issues come from internal IT data administration. Separation of duties is commonly used in large IT organizations so that no single person is in a position to introduce fraudulent or malicious code or have access to restricted data without detection.

In this scenario, we show that it is possible for DBAs to perform their duties without SYSADM authority. In this scenario, a DBA can create and manipulate objects, and the SECADM authority is responsible for grants and revokes.

Important: You must define a plan to identify and associate authorities for all users needing access to the data of your company, assuming there is a separation of duties.

In this scenario, the IT organization in the company needs:

- A RACF group called SECHEAD that has two associated user IDs (PAOLOR1 and PAOLOR2) connected as SECADM1.
- A role called SECAGENT as SECADM2 with two user IDs (PAOLOR3 and PAOLOR4) managed by a trusted context.

To satisfy these requirements, you must complete the following steps:

1. Change DSNZPARM to the values shown in Example 10-1.

   Example 10-1 Altering DSNZPARM

   ```
   SEPARATE_SECURITY=YES,
   SECADM1_TYPE=AUTHID,
   SECADM1=SECHEAD,
   SECADM2_TYPE=ROLE,
   SECADM2=SECAGENT,
   ```

   If you want to use ROLE on both SECADM1_type and SECADM2_type, the current SYSADM needs to create the role and trusted context *before* you change the DSNZPARM, because after you change DSNZPARM, SYSADM will lose that authority.

2. Issue the SET SYSPARM RELOAD command to validate the new DSNZPARM.
3. If you want to monitor all the new administrative authorities, you should insert audit policies in the SYSAUDITPOLICIES table using the DBADMIN column. The values could be:

   – * Audit all the authorities
   – SYSTEM DBADM
   – DBCTRL

– DBADM
– SECADM
– ACCESSCRTL
– SQLADM
– DBMAINT
– PACKADM
– DATACCESS

Choose to audit all authorities and audit all tables with schema SPF and the tables with names starting with the letter “E “.

Example 10-2 shows how PAOLOR1 (SECADM) defines the policy with an INSERT.

Example 10-2 Audit policies

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
 (AUDITPOLICYNAME, DBADMIN)
 VALUES('POLICY_ADMINISTRATIVE','*');

INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE,
  DB2START)
VALUES('POL_TABLE_OBJECTS','SPF','''E%''','T','C','Y');
```

Column DB2START indicates whether the audit policies should start automatically during DB2 start using the ‘Y’ value, or whether the activation should be done using the following command:

```
START TRACE(AUDIT) AUDTPLCY(POLICY_ADMINISTRATIVE)
```

You cannot specify CLASS or IFCID with AUDTPLCY. Most users have, by default, audit class(1,2,7) started because DB2 provides that audit class. Those classes control:

– Class 1: Access attempts are denied due to inadequate authorization.

– Class 2: Provides explicit GRANT and REVOKE.

– Class 7: Assignment or change of authorization ID.

If they are not started yet, you need to choose which classes should be started.

Only the SECADM authority can insert rows in SYSIBM.SYSAUDITPOLICIES. Note that for the column objectname that you need to use three quotation marks to identify the table name when you are using a LIKE predicate. If you are identifying a specific table, you just need to use one quotation marks.

Example 10-3 shows a DISPLAY TRACE command and its output. You must provide the audit policy name in the DISPLAY command.

Example 10-3 DISPLAY TRACE output for POLICY_ADMINISTRATIVE policy

```
DIS TRACE(AUDIT) AUDTPLCY(POLICY_ADMINISTRATIVE) DETAIL(1,2)

DSNW143I  -DBOA CURRENT TRACE QUALIFICATIONS ARE -
DSNW152I  -DBOA BEGIN TNO 03 QUALIFICATIONS:  515
NO QUALIFICATIONS
END TNO 03 QUALIFICATIONS
DSNW185I  -DBOA BEGIN TNO 03 AUDIT POLICIES:  516
ACTIVE AUDIT POLICY: POLICY_ADMINISTRATIVE
END TNO 03 AUDIT POLICIES
DSNW148I  -DBOA ******END OF DISPLAY TRACE QUALIFICATION DATA
```

Tip: You should use the detail option for DISPLAY TRACE to identify the correct trace that you need to stop because all traces started by audit policy appear as: AUDIT * only.

You can activate the policy by running the following command:

```
-STA TRACE(AUDIT) DEST(GTF) AUDTPLCY(POL_TABLE_OBJECTS)
```

Example 10-4 shows the output of POL_TABLE_OBJECTS.

Example 10-4 Display trace output for POL_TABLE_OBJECTS policy

```
DIS TRACE(AUDIT) AUDTPLCY(POL_TABLE_OBJECTS) DETAIL(1,2)

DSNW127I  -DBOA CURRENT TRACE ACTIVITY IS -
TNO TYPE   CLASS        DEST QUAL IFCID
03  AUDIT  *            GTF  NO
*********END OF DISPLAY TRACE SUMMARY DATA*********
DSNW143I  -DBOA CURRENT TRACE QUALIFICATIONS ARE -
DSNW152I  -DBOA BEGIN TNO 03 QUALIFICATIONS:
NO QUALIFICATIONS
END TNO 03 QUALIFICATIONS
DSNW185I  -DBOA BEGIN TNO 03 AUDIT POLICIES:
ACTIVE AUDIT POLICY: POL_TABLE_OBJECTS
END TNO 03 AUDIT POLICIES
DSNW148I  -DBOA ******END OF DISPLAY TRACE QUALIFICATION DATA******
DSN9022I  -DBOA DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION
```

4. Example 10-5 shows how to create role SECAGENT with the SECADM authority by using the SECHEAD RACF group.

Example 10-5 Creating ROLE

```
SET CURRENT SQLID = 'SECHEAD';
---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

CREATE ROLE SECAGENT;
---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+--
```

5. Example 10-6 shows how we create trusted context for the SECAGENT role for user IDs PAOLOR3 and PAOLOR4.

Example 10-6 Creating a trusted context

```
SET CURRENT SQLID = 'SECHEAD';
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
CREATE TRUSTED CONTEXT TXT_SECAGENT
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR4
DEFAULT ROLE SECAGENT WITH ROLE AS OBJECT OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLOR4') ENABLE;
---------+---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-------
```

```
--
  CREATE TRUSTED CONTEXT TXT1_SECAGENT
  BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR3
  DEFAULT ROLE SECAGENT WITH ROLE AS OBJECT OWNER AND QUALIFIER
  ATTRIBUTES (ADDRESS '9.30.28.115') ENABLE;
```

PAOLOR4 logs on to TSO with job PAOLOR4, a trusted connection is established, and PAOLOR4 acquires the SECAGENT role, which has the SECADM authority.

Now PAOLOR4 has SECADM authority and can insert rows on SYSIBM.SYSAUDITPOLICIES, as you can see in Example 10-7. SET CURRENT SQLID is not necessary for this task.

Example 10-7 Insert on SYSIBM.SYSPOLICIES

```
DELETE FROM SYSIBM.SYSAUDITPOLICIES
   WHERE AUDITPOLICYNAME = 'POL_TABLE_OBJECTS';
INSERT INTO SYSIBM.SYSAUDITPOLICIES
  (AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
  VALUES('POL_TEST_I_U_D','SPF','EMP_OTHER_INF','T','C');
---------+---------+---------+---------+---------+---------+---------+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+
```

When we try to execute the same command remotely using PAOLOR4, the message in Example 10-8 displays because there is no trusted context for this user ID using an IP address.

Example 10-8 Insert on SYSIBM.SYSPOLICIES - Remotely from PAOLOR4

```
Error during Prepare
 42501(-551)[IBM][CLI Driver][DB2] SQL0551N  "PAOLOR4" does not have the
required authorization or privilege to perform operation "INSERT" on object
"SYSIBM.SYSAUDITPOLICIES".  SQLSTATE=42501
```

Example 10-9 shows that the insert works for user ID PAOLOR3 because PAOLOR3 has trusted context through an IP address.

Now we have two audit policies:

– One created by PAOLOR4 using SECAGENT ROLE for table EMP_OTHER_INF

– One created by PAOLOR3 using SECAGENT ROLE by IP ADDRESS for table EMP

Example 10-9 Insert remotely using PAOLOR3 user ID

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
  (AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
  VALUES('POL_TEST_I_U_D','SPF','EMP','T','C');

SQL executed OK. Number of rows affected=1 (0.22 secs)
```

6. Create a role and a trusted context for a DBA user ID to work, as shown in Example 10-10.

Example 10-10 Role and Trusted context for DBA user ID

```
SET CURRENT SQLID = 'SECHEAD';
---------+---------+---------+---------+---------+---------+----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+----
```

```
  COMMIT;
---------+---------+---------+---------+---------+---------+----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+----
  CREATE ROLE ROLE_DBA;
---------+---------+---------+---------+---------+---------+----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+----
CREATE TRUSTED CONTEXT TXT_DBA
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR5
DEFAULT ROLE ROLE_DBA WITH ROLE AS OBJECT OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLOR5',JOBNAME 'DB0ACOPY') ENABLE;
---------+---------+---------+---------+---------+---------+----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+----
```

Tip: You should define more than one JOBNAME to take care of your objects. JOBNAME ‘PAOLOR5’ is defined for TSO jobs and JOBNAME ‘DB0ACOPY’is defined for a DB2 maintenance batch job.

7. To create objects, the DBA needs authority through a role. The role should be granted DBADM on SYSTEM by any SECADM, as shown on Example 10-11.

Example 10-11 Grants for DBA user ID

```
GRANT DBADM WITHOUT ACCESSCTRL WITHOUT DATAACCESS TO ROLE ROLE_DBA
GRANT CREATESG TO ROLE ROLE_DBA;
```

8. After those grants, the DBA, PAOLOR5, can work on creating DB2 objects such as stogroup, database, table space, and so on. Example 10-12 shows examples of some objects being created.

Example 10-12 Creating objects

```
CREATE STOGROUP SGR1
    VOLUMES(SBOX80)
    VCAT DBOCD ;
---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-
  COMMIT;
---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-
--
  CREATE DATABASE DBR1
    BUFFERPOOL BP2
    INDEXBP    BP1
    CCSID      EBCDIC
    STOGROUP   SGR1;
---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-
  COMMIT;
---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

```
---------+---------+---------+---------+---------+---------+-
--
  CREATE TABLESPACE TSR1
    IN DBR1 USING STOGROUP SGR1
    PRIQTY 40 SECQTY 40 ERASE NO
    FREEPAGE 0 PCTFREE 5
    GBPCACHE CHANGED TRACKMOD YES
    LOGGED SEGSIZE 4
    BUFFERPOOL BP0 LOCKSIZE ROW
    LOCKMAX 0 CLOSE YES COMPRESS NO
    CCSID EBCDIC DEFINE YES
    MAXROWS 255;
---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+-
   COMMIT;
---------+---------+---------+---------+---------+---------+-
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
CREATE TABLE SPF.EMP_OTHER_INF
      (DEPTNO CHAR(3) FOR SBCS DATA NOT NULL,
       SOCIAL_SECURITY      INTEGER WITH DEFAULT NULL,
       PASSPORT_NUMBER      CHAR(10) FOR SBCS DATA NOT NULL,
       DRIVE_LICENSE        CHAR(10) FOR SBCS DATA NOT NULL,
       USERID               CHAR(7) FOR SBCS DATA NOT NULL,
       CONSTRAINT EMPNO
       PRIMARY KEY (EMPNO))
     IN DBR1.TSR1
     AUDIT NONE DATA CAPTURE NONE
     CCSID EBCDIC NOT VOLATILE APPEND NO;
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

9. Some rows are inserted in the newly created table SPF.EMP_OTHER_INF. Because of separation of duties, the SECADM authority cannot see the data in that table, as shown in Example 10-13.

Example 10-13 Error reading table for SECADM through PAOLOR1

```
SELECT * FROM SPF.EMP_OTHER_INF;
SQLCODE : -551                          DSNTIAR CODE :  0

 DSNT408I SQLCODE = -551, ERROR:  PAOLOR1 DOES NOT HAVE THE PRIVILEGE TO
PERFORM
          OPERATION SELECT ON OBJECT SPF.EMP_OTHER_INF
```

The only people that can see data are DBA PAOLOR5 or SYSADM, as shown in Example 10-14.

Example 10-14 Output for EMP_OTHER_INF table for DBA

```
EMPNO  SOCIAL_SECURITY PASSPORT_NUMBER DRIVE_LICENSE USERID
 ------ --------------- --------------- ------------- -------
4828         123456789 CX34567         X67889676     PAOLOR2
967          768998377 CX78658         X63776234     PAOLOR5
6934         744576861 GH81899         J77436572     PAOLOR3
```

```
3073         983667211 XT11256         L86674532     PAOLOR4
9040         112334555 LW65766         P77688911     PAOLOR1
```

The SYSTEM DBADM authority was granted to the role DBA_ROLE without DATAACCESS, but PAOLOR5 can see the data, as explained in 3.4, “SYSTEM DBADM” on page 42.

Regardless of the SEPARATE_SECURITY setting, an authorization ID or role with the SYSADM authority can use all the privileges of DBADM over any database. If SEPARATE_SECURITY is set to NO, it can also grant other IDs the required privileges to perform the same actions.

In Example 10-13 on page 217, SPF is the schema name, created by the PAOLOR5 user ID, and ROLE_DBA is the owner of the table. ROLE_DBA (owner) is available through a trusted context associated to user ID PAOLOR5.

The SYSTEM DBADM authority was granted to the DBA_ROLE role without DATAACCESS, but PAOLOR5 can see the data, as explained in 3.4, “SYSTEM DBADM” on page 42. PAOLOR5 is able to see the data because PAOLOR5 owns the DBR1 database. If PAOLOR5 created the same SPF.EMP_OTHER_INF table in a database that he does not own, then PAOLOR5 would not be able to see the data, even though he created the table. The owner of the table is SPF. So, it is possible for the SYSTEM DBADM to create a table for someone else and not to be able to access the table.

You cannot prevent PAOLOR5 (DBA) from seeing the data in the table he just created and you cannot prevent the DBA from accessing the data by creating another role and associating PAOLOR5 to the trusted context by executing an ALTER TRUSTED CONTEXT command. If you decide that PAOLOR5 should not see the data, you need to create a row permission that denies the data to PAOLOR5 (DBA).

With the new DB2 administrative authorities, you can reduce the security risks by granting the SYSADM authority to as few users as possible. And with SEPARATE SECURITY = YES, the user ID that was granted SYSADM can no longer control access or manage security-related objects such as roles, trusted context, row permissions, and column masks; only SECADM can do it.

For operation or production activities of the RACF group team, grant specific authorizations such DISPLAYDB, IMAGECOPY, LOAD, REORG, STARTDB, STATS, and STOPDB.

10. As shown in the previous example, there are some tables that should be protected from any unauthorized access. You can create some masks to prevent this situation.

    Going back to the SPF.EMP table, for which a mask was created on SALARY in Example 5-26 on page 101, if you have a mask over any one column, when you try to execute the insert statement shown in Example 10-15, you could receive SQLCODE (-20478), a warning that the definition of the mask conflicts with the requested statement.

Example 10-15 Insert statement

```
INSERT INTO SPF.EMP (SELECT * FROM DSN81010.EMP)

    SQLCODE : -20478                          DSNTIAR CODE :  0

 DSNT408I SQLCODE = -20478, ERROR:  THE STATEMENT CANNOT BE PROCESSED BECAUSE
          COLUMN MASK SALARY_COLUMN_HIDE (DEFINED FOR COLUMN HIREDATE) EXISTS
          AND THE COLUMN MASK CANNOT BE APPLIED OR THE DEFINITION OF THE MASK
          CONFLICTS WITH THE REQUESTED STATEMENT. REASON CODE 30
```

11.Example 10-16 shows a mask on the SPF.EMP_OTHER_INF table that inhibits rows for user IDs that not are connected to the Human Resource RACF group called HRITSO. This mask should be created by the SECADM authority.

Example 10-16 Creating a mask

```
CREATE MASK EMP_OTHERS_MASK ON SPF.EMP_OTHER_INF
     FOR COLUMN SOCIAL_SECURITY RETURN
        CASE WHEN(VERIFY_GROUP_FOR_USER(SESSION_USER,'HRITSO')=1)
             THEN SOCIAL_SECURITY
             ELSE NULL
        END
  ENABLE;
```

Using the SECADM authority, you can activate this column access control by running:

```
ALTER TABLE SPF.EMP_OTHER_INF ACTIVATE COLUMN ACCESS CONTROL;
```

Running a SELECT on the SPF.EMP_OTHER_INF table using PAOLOR7 that is not in HRITSO or using DB2R2 that was granted SYSADM, we have the output shown in Example 10-17.

Example 10-17 Output from SELECT with mask on SOCIAL_SECURITY

```
EMPNO  SOCIAL_SECURITY PASSPORT_NUMBER DRIVE_LICENSE
 ------ --------------- --------------- -------------
 000010              ? CX34567         123456789
 000020              ? CX78658         X63776234
 000030              ? GH81899         J77436572
 000050              ? XT11256         L86674532
 000060              ? LW65766         P77688911
 000070              ? HJ83721         L88576631
 000090              ? PL16234         P27384662
 000100              ? PW27781         K87630029
```

12.The following rules apply to the Spiffy Computer Company:

– PAOLOR8 could list rows just for his own department (A00) on the SPF.EMP_OTHER_INF table.

– PAOLOR8 can select just his own salary from the SPF.EMP table.

– Second line manager PAOLOR6, connected to the MANAGER RACF group, can see data for his own departments (WORKDEPT D21 and E21). To apply these security company rules, we have to create the objects shown in Example 10-18.

Example 10-18 Objects to satisfy the company rules

```
CREATE ROLE ROLE_SPEC_ROW;
CREATE ROLE ROLE_MANAGER;
CREATE TRUSTED CONTEXT TXT_SPEC_ROW
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR8
DEFAULT ROLE ROLE_SPEC_ROW WITH ROLE AS OBJECT OWNER AND QUALIFIER
ATTRIBUTES (JOBNAME 'PAOLOR8',JOBNAME 'DB0ABTCH') ENABLE;

CREATE PERMISSION ROW_EMP_OTHER ON SPF.EMP_OTHER_INF
   FOR ROWS WHERE ((VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(
                  SESSION_USER,'ROLE_SPEC_ROW') = 1 AND
                  DEPTNO = 'A00') OR
                 (VERIFY_GROUP_FOR_USER(SESSION_USER, 'MANAGER') = 1
                  AND DEPTNO IN('D21', 'E21')))
     ENFORCED FOR ALL ACCESS ENABLE;
  --
   CREATE PERMISSION ROW_EMP ON SPF.EMP
   FOR ROWS WHERE (VERIFY_GROUP_FOR_USER(
                  SESSION_USER,'MANAGER') = 1 AND
                  WORKDEPT IN ('D21','E21') OR
                  EMPNO = SESSION_USER)
     ENFORCED FOR ALL ACCESS ENABLE;
```

13. Example 10-19 shows queries executed against the SPF.EMP and SPF.EMP_OTHER_INF tables using both the PAOLOR6 and PAOLOR8 user IDs. Both row permissions should be activated to get these results.

Example 10-19 Results for queries

```
Using PAOLOR8 userid :
---------+---------+---------+---------+---------+---------+---------+---------+
   SELECT * FROM SPF.EMP;
---------+---------+---------+---------+---------+---------+---------+---------+
EMPNO    FIRSTNME      MIDINIT  LASTNAME         WORKDEPT  PHONENO  HIREDATE
---------+---------+---------+---------+---------+---------+---------+---------+
PAOLOR8  CHISTINE      I        HAAS             A00       3978     1965-01-01
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
   SELECT * FROM SPF.EMP_OTHER_INF;
---------+---------+---------+---------+---------+---------+---------+---------+
DEPTNO  SOCIAL_SECURITY  PASSPORT_NUMBER  DRIVE_LICENSE
---------+---------+---------+---------+---------+---------+---------+---------+
A00           123456789  CX34567          X67889676
A00           674453399  IU47224          R43839392
A00           667448222  SJ82781          D82764229
DSNE610I NUMBER OF ROWS DISPLAYED IS 3

Using PAOLOR6 - Manager RACF group
---------+---------+---------+---------+---------+---------+---------+--
   SELECT * FROM SPF.EMP;
---------+---------+---------+---------+---------+---------+---------+--
EMPNO    FIRSTNME      MIDINIT  LASTNAME         WORKDEPT  PHONENO  HIRE
---------+---------+---------+---------+---------+---------+---------+--
000070   EVA           D        PULASKI          D21       7831     1980
000100   THEODORE      Q        SPENSER          E21       0972     1980
000230   JAMES         J        JEFFERSON        D21       4200     1966
000240   SALVATORE M        MARINO           D21       3780     1979
000270   MARIA         L        PEREZ            D21       9001     1980
000320   RAMLAL        V        MEHTA            E21       9990     1965
000330   WING                   LEE              E21       2103     1976
DSNE610I NUMBER OF ROWS DISPLAYED IS 7
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+--
   SELECT * FROM SPF.EMP_OTHER_INF;
---------+---------+---------+---------+---------+---------+---------+--
DEPTNO  SOCIAL_SECURITY  PASSPORT_NUMBER  DRIVE_LICENSE
---------+---------+---------+---------+---------+---------+---------+--
D21           744576861  GH81899          J77436572
E21           982222199  GT87214          Y27834492
E21           292834222  QO33451          T46728029
E21           928457563  TU39911          I37836129
```

```
E21          384756479  NH33994         J49284922
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
```

14. We have created a row permission showing that PAOLOR6, who is in the MANAGER RACF group, could only see rows for his own departments, that is, D21 and E21. If he tries to insert data in that table, the answers for those insert would depend on the department value and would be as shown in Example 10-20.

Example 10-20 Insert on EMP_OTHER_INF table

```
---------+---------+---------+---------+---------+---------+---------+---------+
  INSERT INTO SPF.EMP_OTHER_INF VALUES('D21',886227739,
  'XX18721','T55636666');
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
  COMMIT;
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
  INSERT INTO SPF.EMP_OTHER_INF VALUES('A00',558333333,
  'LO52511','P99447474');
---------+---------+---------+---------+---------+---------+---------+---------+
DSNT408I SQLCODE = -20471, ERROR:  THE INSERT OR UPDATE IS NOT ALLOWED BECAUSE
         A RESULTING ROW DOES NOT SATISFY ROW PERMISSIONS
```

For the update statement, the result should be rc = +100 for departments different from D21 or E21. In Example 10-21, PAOLOR6 tries to update a row for department J22.

Example 10-21 Updating table

```
---------+---------+---------+---------+---------+---------+-----
  UPDATE SPF.EMP_OTHER_INF SET SOCIAL_SECURITY = 222222222 WHERE
  SOCIAL_SECURITY = 493299222;
---------+---------+---------+---------+---------+---------+-----
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

10.3 Scenario 2: Classification of users

In this scenario, we define authorities for users belonging to two different departments: Payroll and IT.

For the Payroll department:

- We have the *Clerk* RACF group that can insert data into the EMP table for the first time, but Clerk cannot see the salary column when executing SELECT from the table.
- The supervisor RACF group is the only group that can update the salary.

For the IT department:

- All employees can only see their own row of data.
- The Administrative RACF group can see any row but not the salary column.
- The Manager RACF group can see all columns from the table but just for their own departments.
- The Director RACF group can see all the data in the table.

Example 10-22 shows a snippet of the select output from the EMP table, with all the columns being representative of user IDs and jobs.

Example 10-22 EMP output data

```
SELECT EMPNO,FIRSTNME,WORKDEPT,JOB,SALARY,BONUS
   FROM SPF.EMP ORDER BY EMPNO;
---------+---------+---------+---------+---------+---------+---------+---------
EMPNO    FIRSTNME      WORKDEPT  JOB           SALARY        BONUS
---------+---------+---------+---------+---------+---------+---------+---------
PAOLOR3  EILEEN        E11       PROJL       29750.00       600.00
PAOLOR5  SEAN          A00       CLERK       29250.00       600.00
PAOLOR6  EVA           D21       MANAGER     36170.00       700.00
PAOLOR7  HEATHER       C01       SUPERV      28420.00       600.00
PAOLOR8  CHISTINE      A00       DIRECTOR    52770.00      1000.00
PAOLOR9  MASATOSHI     D11       DESIGNER    24680.00       500.00
000020   MICHAEL       B01       MANAGER     41250.00       800.00
000030   SALLY         C01       MANAGER     38250.00       800.00
000050   JOHN          E01       MANAGER     40175.00       800.00
```

Complete the following steps:

1. All necessary permissions should be granted to a specific RACF group for payroll employees:
   - CLERK: SELECT, INSERT on the SPF.EMP table.
   - SUPERV: SELECT, UPDATE on the SPF.EMP table.
   - MANAGER, DIRECTOR, PROJL, DESIGNER: SELECT on the SPF.EMP table.
2. Define CREATE statements for row permissions and masks to satisfy the above conditions, as shown in Example 10-23.

Example 10-23 Row permissions and masks definition

```
CREATE PERMISSION SECHEAD.ROW_EMP_RULES ON SPF.EMP
   FOR ROWS
   WHERE (VERIFY_GROUP_FOR_USER(SESSION_USER, 'CLERK') = 1
      OR VERIFY_GROUP_FOR_USER(SESSION_USER, 'PROJL') = 1
      OR VERIFY_GROUP_FOR_USER(SESSION_USER, 'SUPERV') = 1
      OR VERIFY_GROUP_FOR_USER(SESSION_USER, 'MANAGER') = 1
     AND WORKDEPT IN('D21', 'E21')
      OR EMPNO = SESSION_USER)
   ENFORCED FOR ALL ACCESS
   ENABLE ;

CREATE MASK SECHEAD.SALARY_COLUMN_MASK ON SPF.EMP
  FOR COLUMN SALARY RETURN
  CASE
    WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER, 'PROJL') = 1  OR
   VERIFY_GROUP_FOR_USER(SESSION_USER, 'CLERK') = 1) THEN
      NULL
    ELSE
      SALARY
  END
  ENABLE ;
```

3. Example 10-24 shows the results when selecting from the SPF.EMP table with different user IDs. User IDs and RACF groups are the same as shown on Example 10-22 on page 222 for column EMPNO and column JOB.

Example 10-24 Output from select

```
1-) For PAOLOR3 (PROJL racf group) or PAOLOR5 (CLERK racf group)

   EMPNO   FIRSTNME  WORKDEPT JOB           SALARY
*       *         *        *                  *
------- --------- -------- -------- -----------
PAOLOR6 EVA       D21      MANAGER            ?
000100  THEODORE  E21      MANAGER            ?
000230  JAMES     D21      CLERK              ?
000240  SALVATORE D21      CLERK              ?
000250  DANIEL    D21      CLERK              ?
000260  SYBIL     D21      CLERK              ?
000270  MARIA     D21      CLERK              ?
000320  RAMLAL    E21      FIELDREP           ?
000330  WING      E21      FIELDREP           ?
000340  JASON     E21      FIELDREP           ?
200240  ROBERT    D21      CLERK              ?
200330  HELENA    E21      FIELDREP           ?
200340  ROY       E21      FIELDREP           ?

2-) For PAOLOR9 (DESIGNER racf group)

No rows are returned because DESIGNER has authority only for WORKDEPT = D11


3-) For PAOLOR6 (MANAGER racf group)

    EMPNO   FIRSTNME  WORKDEPT JOB           SALARY       BONUS
*       *         *        *                  *           *
------- --------- -------- -------- ----------- -----------
PAOLOR6 EVA       D21      MANAGER     36170.00      700.00
000100  THEODORE  E21      MANAGER     26150.00      500.00
000230  JAMES     D21      CLERK       22180.00      400.00
000240  SALVATORE D21      CLERK       28760.00      600.00
000250  DANIEL    D21      CLERK       19180.00      400.00
000260  SYBIL     D21      CLERK       17250.00      300.00
000270  MARIA     D21      CLERK       27380.00      500.00
000320  RAMLAL    E21      FIELDREP    19950.00      400.00
000330  WING      E21      FIELDREP    25370.00      500.00
000340  JASON     E21      FIELDREP    23840.00      500.00
200240  ROBERT    D21      CLERK       28760.00      600.00
200330  HELENA    E21      FIELDREP    25370.00      500.00
200340  ROY       E21      FIELDREP    23840.00      500.00

4-) For PAOLOR8 (DIRECTOR RACF group)

EMPNO   FIRSTNME  WORKDEPT JOB           SALARY       BONUS
*       *         *        *                  *           *
------- --------- -------- -------- ----------- -----------
PAOLOR1 ERNIE     B01      MANAGER     41250.00      800.00
PAOLOR3 EILEEN    E11      PROJL       29750.00      600.00
```

```
PAOLOR5 SEAN      A00      CLERK       29250.00      600.00
PAOLOR6 EVA       D21      MANAGER     36170.00      700.00
PAOLOR7 HEATHER   C01      SUPERV      28420.00      600.00
PAOLOR8 CHISTINE  A00      DIRECTOR    52770.00     1000.00
PAOLOR9 MASATOSHI D11      DESIGNER    24680.00      500.00
XXXXXX  CHRISTINE A00      YYYY        52750.00     1000.00
000030  SALLY     C01      MANAGER     38250.00      800.00
000050  JOHN      E01      MANAGER     40175.00      800.00
000060  IRVING    D11      MANAGER     32250.00      600.00
000100  THEODORE  E21      MANAGER     26150.00      500.00
000110  VINCENZO  A00      SALESREP    46500.00      900.00
000130  DOLORES   C01      ANALYST     23800.00      500.00
000150  BRUCE     D11      DESIGNER    25280.00      500.00
000160  ELIZABETH D11      DESIGNER    22250.00      400.00
000180  MARILYN   D11      DESIGNER    21340.00      500.00
000190  JAMES     D11      DESIGNER    20450.00      400.00
000200  DAVID     D11      DESIGNER    27740.00      600.00
000210  WILLIAM   D11      DESIGNER    18270.00      400.00
000220  JENNIFER  D11      DESIGNER    29840.00      600.00
000230  JAMES     D21      CLERK       22180.00      400.00
000240  SALVATORE D21      CLERK       28760.00      600.00
000250  DANIEL    D21      CLERK       19180.00      400.00
000260  SYBIL     D21      CLERK       17250.00      300.00
000270  MARIA     D21      CLERK       27380.00      500.00
000280  ETHEL     E11      OPERATOR    26250.00      500.00
000290  JOHN      E11      OPERATOR    15340.00      300.00
000300  PHILIP    E11      OPERATOR    17750.00      400.00
000310  MAUDE     E11      OPERATOR    15900.00      300.00
000320  RAMLAL    E21      FIELDREP    19950.00      400.00
000330  WING      E21      FIELDREP    25370.00      500.00
000340  JASON     E21      FIELDREP    23840.00      500.00
200010  DIAN      A00      SALESREP    46500.00     1000.00
200120  GREG      A00      CLERK       29250.00      600.00
200140  KIM       C01      ANALYST     28420.00      600.00
```

10.4 The SYSADM authority

The superuser authority is powerful and there is a risk that a person having SYSADM can misuse their authority. For example, such a person can:

- See all the data in the subsystem
- Arrange for one authid to take over the authority of another authid
- Bypass access controls

Security can be improved with SEPARATE_SECURITY set to YES, as the people with SYSADM authority can no longer manage security objects. This setup helps protect business data from insiders. If you have SEPARATE_SECURITY set to NO, consider establishing an audit policy with an audit category of SECMAINT that allows you to create trusted contexts and roles.

We show examples of a DBA getting help from a system administrator to take over the authority of another DBA. Another example shows how a user with SYSADM authority can view data by using a row permission.

The three examples in this section highlight the power of the SYSADM authority when it is used to create trusted contexts and roles.

10.4.1 Setting SEPARATE_SECURITY to YES: Concerns

Prior to changing to an environment with SEPARATE_SECURITY set to YES, you need to be aware of existing trusted contexts and make sure they are justified and protected.

Suppose the new setup has the following settings:

- `SECADM1= SECHEAD` (a RACF group with two members, PAOLOR1 and PAOLOR2)
- `SECADM2=SECAGENT` (a Role)

The subsystem is about to change to SEPARATE_SECURITY to YES. DB2R2 currently has the SYSADM authority and wants to keep it. DB2R2 discovers the names of the authids belonging to RACF group SECHEAD, which are PAOLOR1 and PAOLOR2. In Example 10-25, DB2R2 sets up a trusted context before setting SEPARATE_SECURITY to YES.

Example 10-25 Create a trusted connection

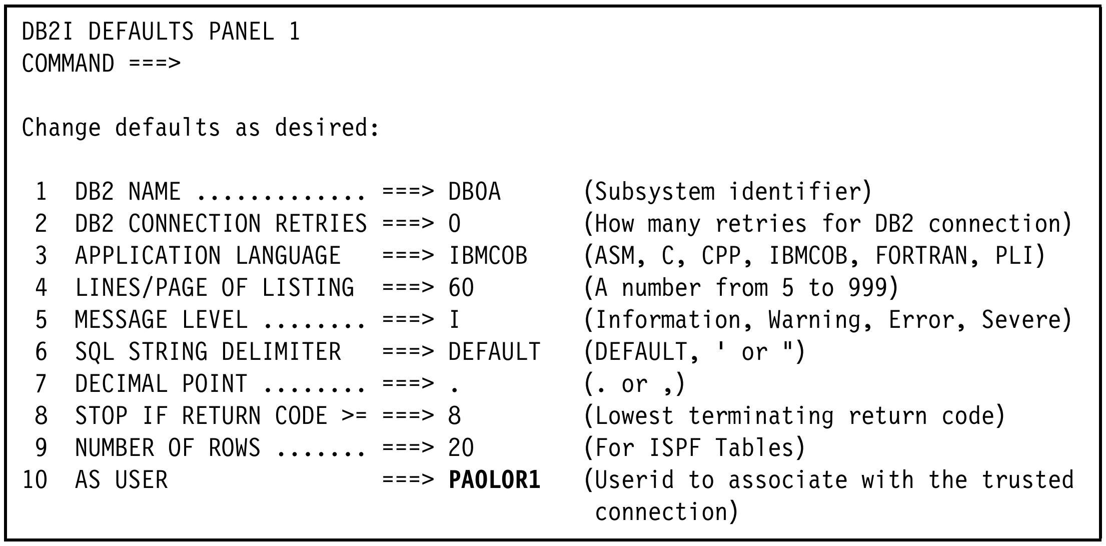
```
CREATE TRUSTED CONTEXT TC_DB2R2_BECOME_SEC_HEAD
BASED UPON CONNECTION USING SYSTEM AUTHID DB2R2
ATTRIBUTES (JOBNAME 'DB2R2')
WITH USE FOR PAOLOR1
ENABLE;
```

The subsystem now has SEPARATE_SECURITY set to YES. DB2R2 can take over the authority of PAOLOR1 and create security related objects. DB2R2 logs on to TSO, and a trusted connection is established, as shown in Example 10-26. The subsystem starts and has SEPARATE_SECURITY set to YES.

Example 10-26 New trusted context

```
V485-TRUSTED CONTEXT=TC_DB2R2_BECOME_SEC_HEAD,
     SYSTEM AUTHID=DB2R2,
     ROLE=*
```

Within the trusted connection, DB2R2 goes to the SPUFI defaults panel and switches to PAOLOR1 in the ASUSER default value, as shown in Figure 10-1.



```
DB2I DEFAULTS PANEL 1
COMMAND ===>

Change defaults as desired:

 1  DB2 NAME ............. ===> DB0A      (Subsystem identifier)
 2  DB2 CONNECTION RETRIES ===> 0         (How many retries for DB2 connection)
 3  APPLICATION LANGUAGE   ===> IBMCOB    (ASM, C, CPP, IBMCOB, FORTRAN, PLI)
 4  LINES/PAGE OF LISTING  ===> 60        (A number from 5 to 999)
 5  MESSAGE LEVEL ........ ===> I         (Information, Warning, Error, Severe)
 6  SQL STRING DELIMITER   ===> DEFAULT   (DEFAULT, ' or ")
 7  DECIMAL POINT ........ ===> .         (. or ,)
 8  STOP IF RETURN CODE >= ===> 8         (Lowest terminating return code)
 9  NUMBER OF ROWS ....... ===> 20        (For ISPF Tables)
10  AS USER                ===> PAOLOR1   (Userid to associate with the trusted
                                           connection)
```

Figure 10-1 Preparing to switch

DB2R2, as PAOLOR1, can now create security-related objects, as shown in Example 10-27.

Example 10-27 Creating a role with AS USER

```
SET CURRENT SQLID='SECHEAD';
---------+---------+---------+---------+---------+---------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------
CREATE ROLE ABC;
---------+---------+---------+---------+---------+---------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------
```

DB2R2 has managed to retain its implicit SECADM authority through a trusted connection.

Carefully examine all existing roles and trusted connection prior to setting SEPARATE_SECURITY to YES. If you want to drop a role, it cannot own any objects.

Once there are no trusted contexts or roles that can acquire SECADM authority through SECADM1 or SECADM2, you can move ahead and set SEPARATE_SECURITY to YES.

10.4.2 Taking over the authority of another person

In this example, there are three application databases in a production environment. Each database is supported by a different DBA, each holding the DBADM privilege over their respective database. Work needs to be done on a particular database and the DBA maintaining that application is not available. We show, using a trusted context, how one DBA can act as another DBA in the event that the other DBA is unavailable. The three databases are:

- DLOAN
- DMTG
- DCARD

The following steps illustrate this scenario:

1. A user ID with SYSADM executes the statements shown in Example 10-28.

Example 10-28 Establish DBA responsibility

```
CREATE STOGROUP SGRB0201
VOLUMES(XTRA01)
VCAT DSNCAT;

CREATE DATABASE DLOAN
    BUFFERPOOL BP1
    STOGROUP SGRB0201;

CREATE DATABASE DMTG
    BUFFERPOOL BP1
    STOGROUP SGRB0201;

CREATE DATABASE DCARD
    BUFFERPOOL BP1
    STOGROUP SGRB0201;
GRANT DBADM ON DATABASE DLOAN TO USRT020;
GRANT DBADM ON DATABASE DMTG  TO USRT030;
GRANT DBADM ON DATABASE DCARD TO USRT040;
GRANT USE OF STOGROUP SGRB0201 TO USRT020;
GRANT USE OF STOGROUP SGRB0201 TO USRT030;
GRANT USE OF STOGROUP SGRB0201 TO USRT040
```

A new customer table needs to be created in the DCARD database. DBA USRT040 is not available.

2. USRT030 executes the DDL shown in Example 10-29.

Example 10-29 Create a CUSTOMER table

```
   CREATE TABLE  CARD.CUSTOMER(
       CUSTID        CHAR(10) NOT NULL PRIMARY KEY,
       CARD_NUMBER   CHAR(16),
       BALANCE       DEC(10,2),
       LIMIT         DEC(10,2)
       )
    IN DATABASE DCARD;
   DSNT408I SQLCODE = -551, ERROR:  CARD DOES NOT HAVE THE PRIVILEGE TO
   PERFORM
         OPERATION CREATE TABLE ON OBJECT DCARD
```

USRT030 is connected to the CARD RACF group.

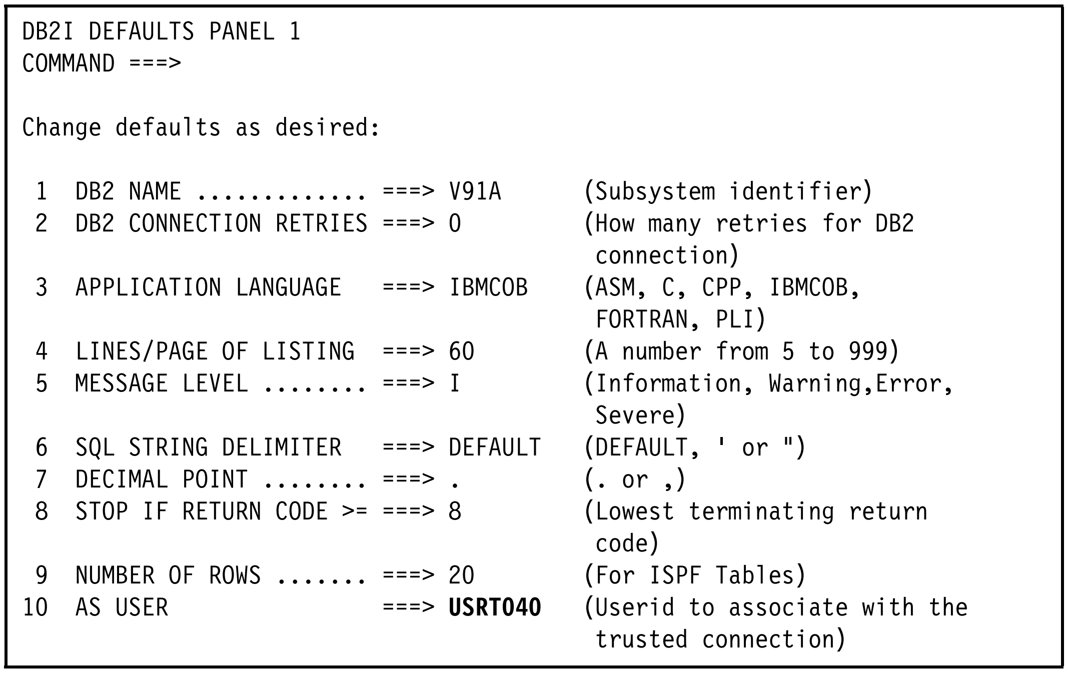
3. A user ID with SYSADM creates the context shown in Example 10-30 to enable USRT030 to perform this urgent implementation.

Example 10-30 Creating a trusted context

```
CREATE TRUSTED CONTEXT CTXT_BACKUP_PROD_DBA
  BASED UPON CONNECTION USING SYSTEM AUTHID USRT030
  ATTRIBUTES (JOBNAME 'USRT030')
  WITH USE FOR USRT040
  ENABLE
  ;
```

   This trusted context allows USRT030 to connect and switch to user ID USRT040 to implement the new table.

4. USRT030 logs on to TSO, which establishes a trusted connection because the SYSTEM AUTHID and job name match the attributes of a defined trusted context. USRT030 opens the DB2I defaults panel shown in Figure 10-2 and sets ASUSER=USRT040.



```
DB2I DEFAULTS PANEL 1
COMMAND ===>

Change defaults as desired:

 1  DB2 NAME ............. ===> V91A      (Subsystem identifier)
 2  DB2 CONNECTION RETRIES ===> 0         (How many retries for DB2
                                           connection)
 3  APPLICATION LANGUAGE   ===> IBMCOB    (ASM, C, CPP, IBMCOB,
                                           FORTRAN, PLI)
 4  LINES/PAGE OF LISTING  ===> 60        (A number from 5 to 999)
 5  MESSAGE LEVEL ........ ===> I         (Information, Warning,Error,
                                           Severe)
 6  SQL STRING DELIMITER   ===> DEFAULT   (DEFAULT, ' or ")
 7  DECIMAL POINT ........ ===> .         (. or ,)
 8  STOP IF RETURN CODE >= ===> 8         (Lowest terminating return
                                           code)
 9  NUMBER OF ROWS ....... ===> 20        (For ISPF Tables)
10  AS USER                ===> USRT040   (Userid to associate with the
                                           trusted connection)
```

Figure 10-2 DSNEOP01 DB2I defaults panel

   Within the established trusted connection, USRT030 is allowed to proceed as USRT040. If IFCID 269 is activated, DB2 records the user switch.

   USRT030 as user USRT040 now holds the DBADM privilege on the DCARD database.

5. USRT030 as USRT040 executes the statements shown in Example 10-31 from SPUFI, The person holding authid USRT030 has now assumed authid USRT040.

Example 10-31 Create a table for an assumed authid

```
CREATE TABLE CARD.CUSTOMER(
   CUSTID CHAR(10) NOT NULL PRIMARY KEY,
   CARD_NUMBER CHAR(16),
   BALANCE DEC(10,2),
   LIMIT DEC(10,2) )
 IN DATABASE DCARD;
```

```
---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+
```

The statement is successful because the DBADM privilege held by USRT040 is used.

In Example 10-32, we see the contents of SYSIBM.SYSTABLES for the new table.

Example 10-32 Customer table information

```
CREATOR=CARD,
NAME=CUSTOMER,
DBNAME=DCARD,
CREATEDBY=USRT040
OWNER=CARD.
```

6. The user ID with SYSADM disables the trusted context:

   ```
   ALTER TRUSTED CONTEXT CTXT_BACKUP_PROD_DBA ALTER DISABLE ;
   ```

7. USRT030 tries to switch to USRT040 on the DB2I defaults panel. The result is:

   ```
   YOU ARE NOT AUTHORIZED TO CONNECT TO DB2 ASUSER USRT040
   ***
   ```

In a situation where an unexpected request occurred, it is possible, using a trusted context, for one DBA to assume the identity of another DBA and create the table that was urgently needed in the production environment. In this example, USRT030 is allowed to run under the authorization ID of USRT040.

This is a powerful capability, and the creation of trusted contexts and roles needs to be carefully controlled and always monitored. If there are many people holding the SYSADM authority, it is a possible vulnerability.

10.4.3 Using a trusted context to circumvent a row permission rule

We show how an access control can be bypassed using an authid with SYSADM privilege by completing the following steps:

1. DB2R9, holding the SYSADM authority, grants PAOLOR9 the SYSTEM DBADM authority:

   ```
   GRANT DBADM ON SYSTEM TO PAOLOR9;
   ```

2. PAOLOR9 creates database D9, table space S9, and table R9.EMPL. Example 10-33 shows the columns for this table.

Example 10-33 Employee table

```
CREATE TABLE
R9.EMPL
     (EMPLID CHAR(6) NOT NULL ,
      DEPTID CHAR (3) NOT NULL,
      SALARY  BIGINT NOT NULL,
      PRIMARY KEY (EMPLID)
     ) IN D9.S9;
```

3. PAOLOR9 populates the table (Table 10-1).

Table 10-1 Data in employee table

EMPLID	DEPTID	SALARY
111111	ACT	50000
222222	ACT	60000
333333	SAL	70000
444444	DEV	80000
555555	SUP	90000
666666	PUR	100000
777777	PUR	110000
888881	ACT	120000
888882	SAL	130000
888883	DEV	140000
999999	FIN	150000

4. DB2R2 holds the implicit SECADM authority and creates a row permission that only allows users with the ‘HRIBM’ role to see the rows in this table. The permission is created and activated, as shown in Example 10-34.

Example 10-34 Create a permission

```
--COMMIT;
  CREATE PERMISSION SALARY_ACCESS ON R9.EMPL
  FOR ROWS WHERE
 ((VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,'HRIBM') = 1))
  ENFORCED FOR ALL ACCESS
  ENABLE;
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+--------
  COMMIT;
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+--------
ALTER TABLE R9.EMPL ACTIVATE ROW ACCESS CONTROL;
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

5. DB2R2 now gives select access on this table to the HRIBM role, as shown in Example 10-35.

Example 10-35 Grant to a role

```
GRANT SELECT ON R9.EMPL TO ROLE HRIBM;
---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

6. In Example 10-36, PAOLOR9 issues a SELECT against the table and sees no rows, even though he created the table.

Example 10-36 Select on R9.EMPL

```
SELECT * FROM R9.EMPL;
---------+---------+---------+---------+---------+---------+
EMPLID  DEPTID                   SALARY
---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

7. In Example 10-37, PAOLOR8, a new HR user, issues a SELECT against the table and is not successful.

Example 10-37 Select on R9.EMPL

```
SELECT * FROM R9.EMPL;
---------+---------+---------+---------+---------+---------+---------+--
DSNT408I SQLCODE = -551, ERROR:  PAOLOR8 DOES NOT HAVE THE PRIVILEGE TO
         PERFORM OPERATION SELECT ON OBJECT R9.EMPL
DSNT418I SQLSTATE   = 42501 SQLSTATE RETURN CODE
```

8. DB2R2 creates a trusted context that allows the PAOLOR8 authid to access the rows in the employee table. The HRIBM role is the default role acquired when PAOLOR8 logs on to TSO. To enable PAOLOR8, DB2R2 issues the DDL shown in Example 10-38.

Example 10-38 Create a role and trusted context

```
CREATE ROLE HRIBM;
CREATE TRUSTED CONTEXT TC_PAOLOR8
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLOR8
DEFAULT ROLE HRIBM
ATTRIBUTES (JOBNAME 'PAOLOR8')
ENABLE;
---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

9. PAOLOR8 logs off of TSO and logs back on, and then issues a SELECT, as shown in Example 10-39.

Example 10-39 PAOLOR8 issues SELECT

```
SELECT * FROM R9.EMPL;
---------+---------+---------+---------+---------+---------
EMPLID  DEPTID                   SALARY
---------+---------+---------+---------+---------+---------
111111  ACT                       50000
222222  ACT                       60000
333333  SAL                       70000
444444  DEV                       80000
555555  SUP                       90000
666666  PUR                      100000
777777  PUR                      110000
888881  ACT                      120000
888882  SAL                      130000
888883  DEV                      140000
999999  FIN                      150000
```

```
DSNE610I NUMBER OF ROWS DISPLAYED IS 11
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

10. DB2R2 issues a DISPLAY THREAD command, as shown in Example 10-40, to see the active trusted connection.

Example 10-40 See active trusted connection

```
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
RRSAF    T       20 DBOCADMT_II  STC      ?RRSAF   008B     2
RRSAF    T    25620 DBOCADMT_DMN STC      ?RRSAF   008B     3
TSO      N       25 PAOLOR8      PAOLOR8           0134     0
V485-TRUSTED CONTEXT=TC_PAOLOR8,
     SYSTEM AUTHID=PAOLOR8,
     ROLE=HRIBM
TSO      T  *     3 DB2R2        DB2R2             00FA  2126
TSO      N       45 PAOLOR9      PAOLOR9           0132     0
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -DBOC DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
***
```

11. Another administrator, DB2R9 (SYSADM), is not authorized to see the employee salaries, but wants to access the data in this table without permission. DB2R9 issues a SELECT, as shown in Example 10-41.

Example 10-41 DB2R9 tries to see salary data

```
SELECT * FROM R9.EMPL;
---------+---------+---------+---------+---------+---------+
EMPLID  DEPTID                 SALARY
---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

Although DB2R9 (SYSADM) has access to the table, the user sees no rows. There is a row permission defined on this table, and DB2R9 does not satisfy the row access control rule for the table. In this case, only an authid holding the HRIBM role can see all the rows in the table.

12. Now DB2R9 maliciously assigns himself a role that allows access to the data. He creates a trusted context, as shown in Example 10-42.

Example 10-42 DB2R9 creates a trusted context to see salaries

```
CREATE TRUSTED CONTEXT TC_DB2R9
  BASED UPON CONNECTION USING SYSTEM AUTHID DB2R9
  DEFAULT ROLE HRIBM
  ATTRIBUTES (JOBNAME 'DB2R9')
  ENABLE;
---------+---------+---------+---------+---------+-------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

13. DB2R9 logs off of TSO and logs back in. He first displays the trusted connection to confirm that he has acquired the HRIBM role, as shown in Example 10-43.

Example 10-43 Display trusted connections

```
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
RRSAF    T       20 DBOCADMT_II  STC      ?RRSAF   008B     2
RRSAF    T    25665 DBOCADMT_DMN STC      ?RRSAF   008B     3
TSO      N       25 PAOLOR8      PAOLOR8           0134     0
V485-TRUSTED CONTEXT=TC_PAOLOR8,
     SYSTEM AUTHID=PAOLOR8,
     ROLE=HRIBM
TSO      T  *     3 DB2R9        DB2R9             0132  2136
V485-TRUSTED CONTEXT=TC_DB2R9,
     SYSTEM AUTHID=DB2R9,
     ROLE=HRIBM
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -DBOC DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
***
```

We see that both PAOLOR8 and DB2R9 currently hold the HRIBM role.

14. DB2R9 now issues a select against the employee table, as shown in Example 10-44.

Example 10-44 Data in employee table

```
SELECT * FROM R9.EMPL;
---------+---------+---------+---------+---------+---------+-
EMPLID  DEPTID                 SALARY
---------+---------+---------+---------+---------+---------+-
111111  ACT                     50000
222222  ACT                     60000
333333  SAL                     70000
444444  DEV                     80000
555555  SUP                     90000
666666  PUR                    100000
777777  PUR                    110000
888881  ACT                    120000
888882  SAL                    130000
888883  DEV                    140000
999999  FIN                    150000
DSNE610I NUMBER OF ROWS DISPLAYED IS 11
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

15. DB2R9 removes the trusted context, as shown in Example 10-45.

Example 10-45 Remove the trusted context

```
---------+---------+---------+---------+---------+---------+---
  DROP TRUSTED CONTEXT TC_DB2R9;
---------+---------+---------+---------+---------+---------+---
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

If SEPARATE_SECURITY is set to NO and there are many people holding the SYSADM authority, then there is a potential vulnerability that can lead to insider misuse. In this example, we have shown that a SYSADM can create a trusted context to acquire the role HRIBM and see through row permissions.

11

Remote client applications access

Remote client applications access is now common. Most companies allow connections from the web to do business. The extreme variety of workloads and users has created performance and security issues.

However, business requirements dictate that we allow access from remote clients every time, from everywhere, and the data server needs to integrate new functions to authenticate user identification and verification. That is the reason why we are enabling remote access over a VPN connection and strengthening authentication with stricter rules.

In this chapter, we discuss how DB2 10 for z/OS strengthens authentication for the remote client applications accessing DB2 data over browsers, servers, and virtual private network (VPN) devices.

This chapter contains the following topics:

- Using a password phrase for remote client applications
- Protecting data through DB2 SSL with digital certification
- Identity propagation for a remote client application
- Considerations about SQL injection

These first three functions are available in DB2 10 NFM.

11.1 Using a password phrase for remote client applications

In this scenario, we discuss the appropriate usage of a password phrase.

The first action of system defense is to create the strongest possible password. This is the easiest and most powerful way to protect the company and customers' information assets.

In z/OS V1R8, password phrases were introduced as an alternative to passwords. Password phrases allow a larger character set and also a longer length than normal passwords. Password phrases can be from 14 to 100 characters in length.

With z/OS V1R9, the 14-character minimum length has been relaxed in the interest of interoperability with systems that allow passwords of 9 to 13 characters.

Starting in z/OS V1R10, it is now allowable to specify PASSWRD= and lNEWPHRASE= when PASSWRD= specifies a PassTicket instead of a password.

Let us now explore this topic using a scenario based on the Spiffy Computer Company example.

11.1.1 Background information for the strong password scenario

The Spiffy Computer Company has IBM WebSphere Application Server V7 Network Deployment running on Widows XP and web applications running on DB2 10 for z/OS with z/OS V1.12. Figure 11-1 shows how Spiffy Computer Company's mobile and non-mobile employees can connect to the internal system.

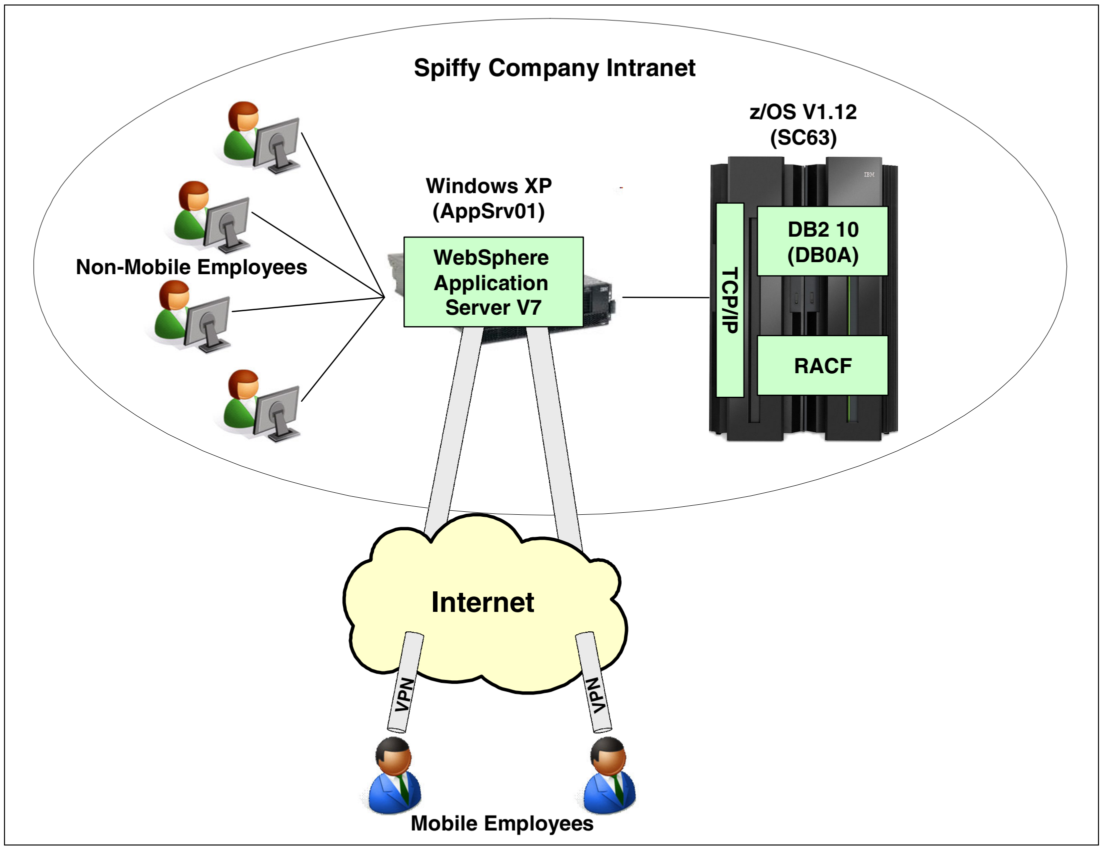


Figure 11-1 Connecting non-mobile and mobile employees of the Spiffy Computer Company

Most employees are working at the office or data center, but all the employee of the Sales and Consulting Department are mobile, working anywhere and anytime. The employees who are working for Sales and Consulting Department access WebSphere Application Server through a VPN service.

After being audited by the internal IT audit team, an audit report highlighted that one mobile employee of the Sales and Consulting Department failed to log on to the internal system after three attempts within 1 minute from a customer site. After some investigation, it turns out that it was done by an unidentified person using the employee's mobile computer.

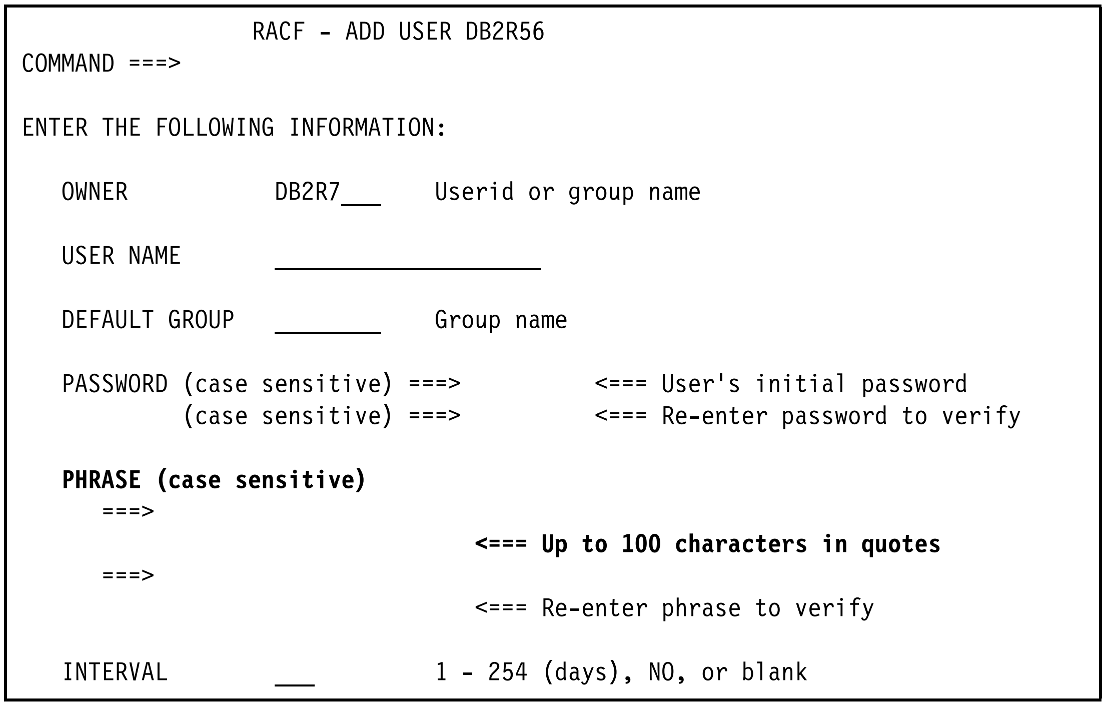
The audit team and the security team decide to change all mobile employees password to password phrase. They believe that longer passwords are a simple but powerful way to keep away phreakers and hackers.

11.1.2 Implementing the password phrase solution

The RACF administrator working for the security team has been asked to change all mobile employees passwords to a password phrase. The RACF administrator can implement this requirement immediately because the password phrase was introduced with z/OS Security Server V1.8 and the current security system is already at Version 1.12.

As an example, the RACF administrator wants to change the password of user ID DB2R56, which belongs to Jimbo, a new employee of the Sales and Consulting team, to a password phrase. So the administrator logs on the system as DB2R7 and makes the change using the RACF panel shown in Figure 11-2.

This is a simple change, and the password phrase is easy to remember. For better security, the phrase should be case sensitive with special characters and include spaces.


```
                 RACF - ADD USER DB2R56
COMMAND ===>

ENTER THE FOLLOWING INFORMATION:

   OWNER           DB2R7___    Userid or group name

   USER NAME       ____________________

   DEFAULT GROUP   ________    Group name

   PASSWORD (case sensitive) ===>         <=== User's initial password
            (case sensitive) ===>         <=== Re-enter password to verify

   PHRASE (case sensitive)
      ===>
                               <=== Up to 100 characters in quotes
      ===>
                               <=== Re-enter phrase to verify

   INTERVAL        ___         1 - 254 (days), NO, or blank
```
Figure 11-2 Changing to a password phrase

After changing DB2R56's password to a password phrase, the RACF administrator sends an email to Jimbo asking him to change the initial password phrase and try to use it. A bit later, the RACF administrator gets an email confirmation from Jimbo that everything is OK, so the RACF administrator can announce to all mobile employees that the password system will be changed the following week and that it should be a password phrase with more than 14 characters.

A few weeks later, the security team announces a new security policy about using a password phrase for WebSphere Application Server applications.

In the current application environment, WebSphere Application Server and DB2 for z/OS are using just one JDBC data source and it is not using a password phrase. This is a security exposure. Figure 11-3 shows us how the applications are currently working and what kind of security problems can occur.

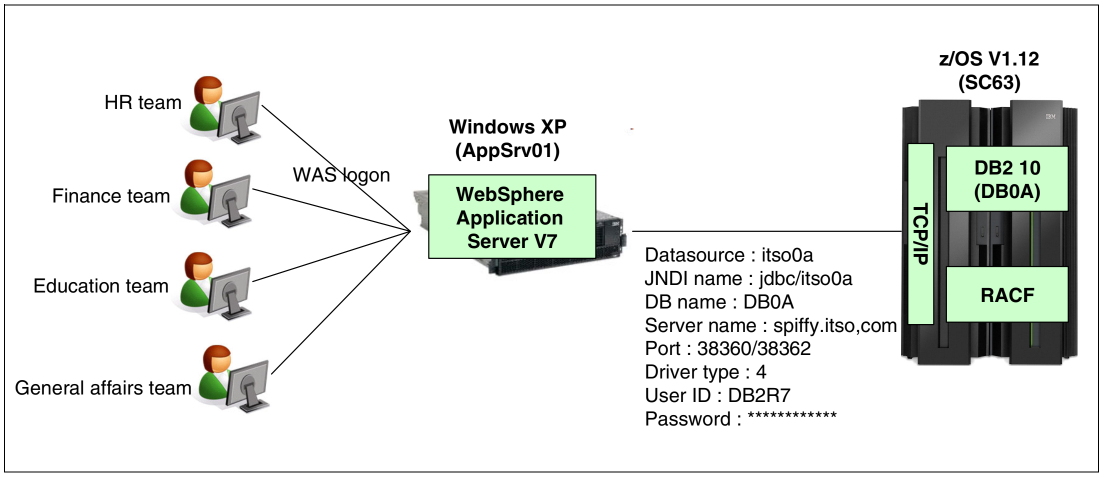


Figure 11-3 Spiffy Computer Company application environment

As the security team mentioned, if the application security (DB2R7 user ID and password) is exposed, it might cause a serious problem.

The security team therefore proposes that a separate data source should be used by each business unit with different user IDs and password phrases. The team subdivides their applications into the following business units: Financial, HR, Common, and Audit. They create data sources for each business unit, named, respectively, itso_dsFinance, itso_dsHR, itso_dsCommon, and itso_dsAudit, as shown in Figure 11-4.

All user IDs and password phrases are different and are given the least privilege to reduce the impact in case of a possible breach.

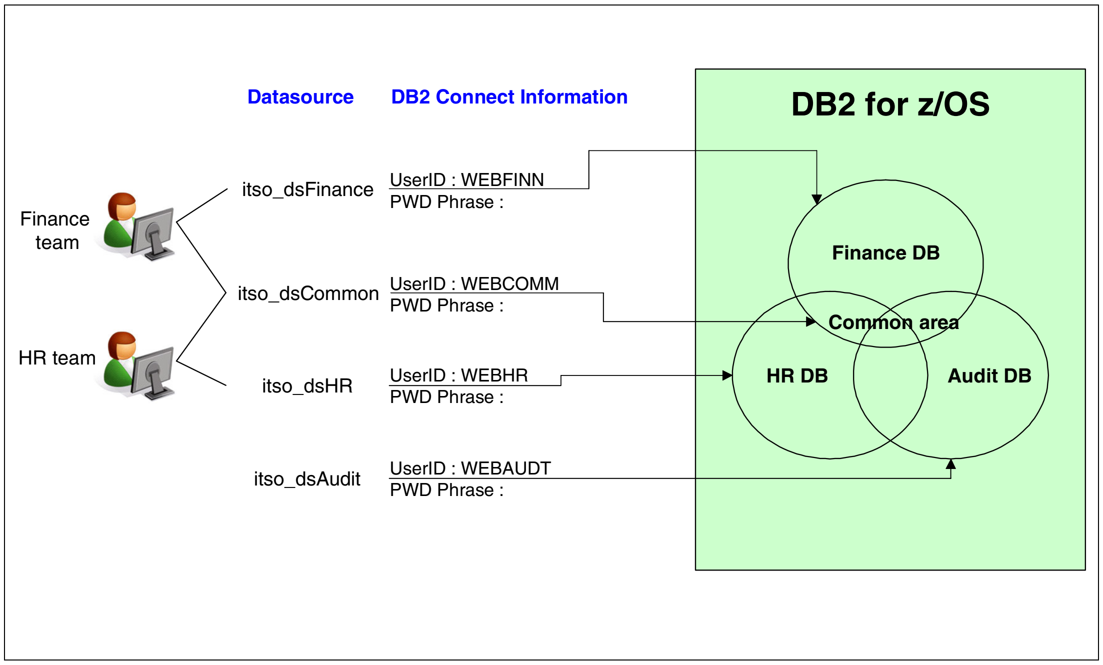
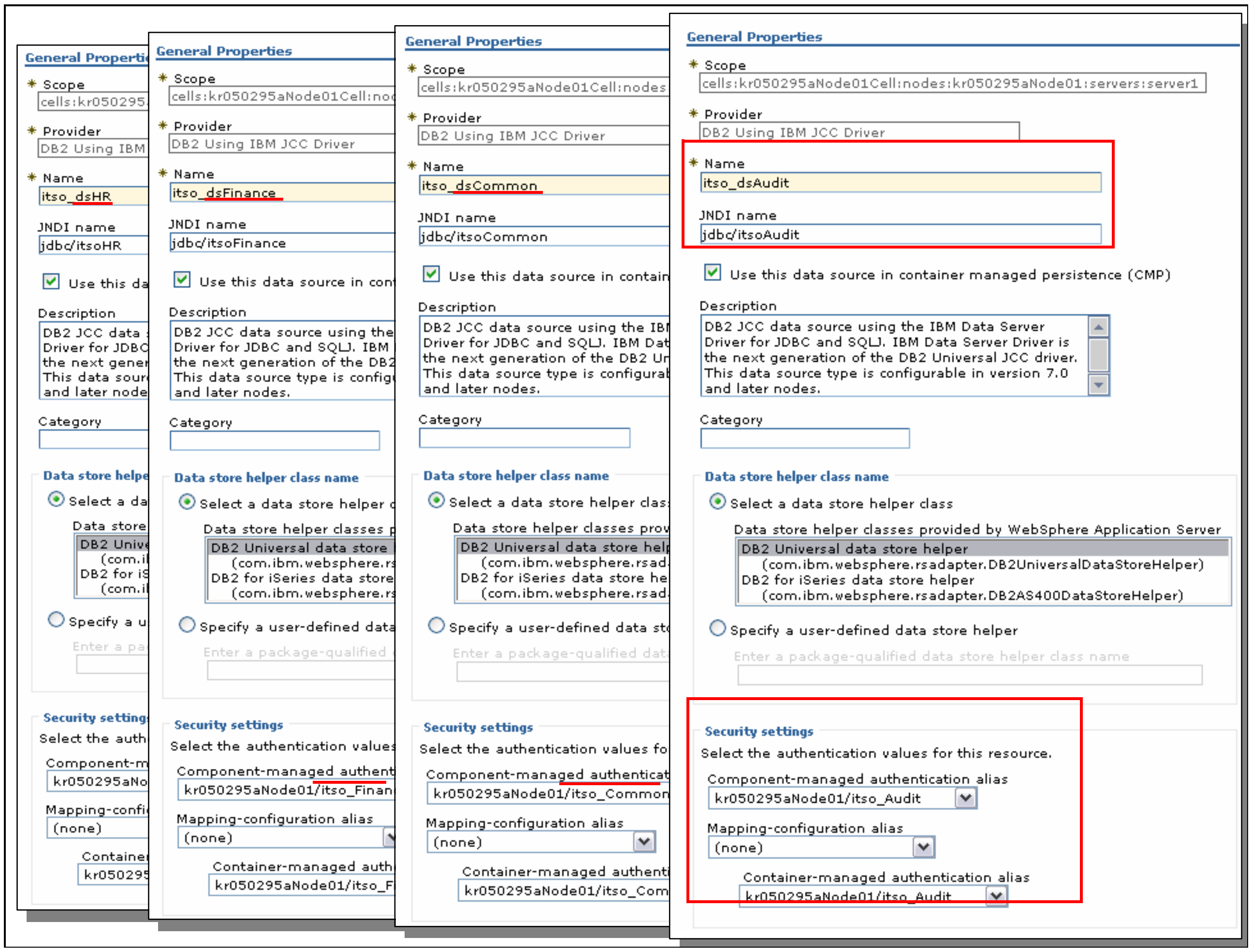


Figure 11-4 Subdividing applications by business unit

Figure 11-5 shows the WebSphere Application Server data sources and its security settings in accordance with the structure described in Figure 11-4 on page 239. The passwords must be replaced by password phrases to meet the security requirement.

Subdividing the Spiffy Computer Company's data sources into the most granular units and giving them the least privilege would reinforce security, but it might cause some management complications. It is appropriate to find the best intermediate point of granularity.



General Properties

* Scope
cells:kr050295aNode01Cell:nodes:kr050295aNode01:servers:server1

* Provider
DB2 Using IBM JCC Driver

* Name
itso_dsAudit

JNDI name
jdbc/itsoAudit

☑ Use this data source in container managed persistence (CMP)

Description
DB2 JCC data source using the IBM Data Server Driver for JDBC and SQLJ. IBM Data Server Driver is the next generation of the DB2 Universal JCC driver. This data source type is configurable in version 7.0 and later nodes.

Category

Data store helper class name

◉ Select a data store helper class

Data store helper classes provided by WebSphere Application Server
DB2 Universal data store helper
(com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper)
DB2 for iSeries data store helper
(com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper)

○ Specify a user-defined data store helper

Enter a package-qualified data store helper class name

Security settings

Select the authentication values for this resource.

Component-managed authentication alias
kr050295aNode01/itso_Audit

Mapping-configuration alias
(none)

Container-managed authentication alias
kr050295aNode01/itso_Audit

Name: itso_dsHR — JNDI name: jdbc/itsoHR
Name: itso_dsFinance — JNDI name: jdbc/itsoFinance — Component-managed authent: kr050295aNode01/itso_Finan
Name: itso_dsCommon — JNDI name: jdbc/itsoCommon — Component-managed authenticat: kr050295aNode01/itso_Common

Figure 11-5 Subdivide JDBC data source and separate security setting with password phrase

Now, Spiffy Computer Company's security is reinforced from the password point of view. Going one step forward, DB2 Connect user's and application's password policy should also be changed to a password phrase in the near future.

11.1.3 Lessons learned

Changing the password policy to a password phrase is an easy solution for a wide spread exposure. Password stealing (phishing attack) is common and it is the first attempt when trying to illegally acquire an authority.

Make sure you use a secure HTTPS connection (indicated by `https://`) when entering user names and passwords, and that the associated certificate is valid.

When using a password phrase, make sure your password policy complies with the following common guidelines:

- Do not share the user ID and password.
- The password must be changed on at least a quarterly basis. But monthly is recommended, especially for system level passwords.
- You must mix uppercase and lowercase characters.
- The password must contain digits.
- The password contain non-alphabetic characters (for example, spaces and special characters).
- Monitor unsuccessful login attempts.
- Avoid recycling old passwords.

And also remember the characteristics of weak passwords:

- The password contains less than fifteen characters.
- The password is a word found in a dictionary (English or foreign).
- The password is a common usage word, such as:
  - The words "<Company Name>", "sanjose", "sanfran" or any derivation
  - Word or number patterns like aaabbb, qwerty, zyxwvuts, 123321, etc.
  - Any of the above preceded or followed by a digit (e.g., helloHR01, 02helloHR)

11.2 Protecting data through DB2 SSL with digital certification

In this scenario, we discuss logging on to DB2 for z/OS from a remote client using z/OS digital certification. We use the DB2 Secure Socket layer (SSL) support to protect Spiffy Computer Company's sensitive data.

The application of public-key technology requires the user of a public key to be confident that the public key belongs to the correct remote person or system with which an encryption or digital signature mechanism is used. This confidence is obtained through the use of public-key certificates. A digital certificate is analogous to a passport. The passport certifies the bearer's identity, address, and citizenship. The concepts behind passports and other identification documents, such as drivers licenses, are similar to those that are used for digital certificates.

Identification documents are issued by a trusted authority, such as the government passport office or a Department of Motor Vehicles. A passport is not issued unless the person who requests it can prove identity and citizenship to the authority. Specialized equipment is used in the creation of passports to make it difficult to alter the information in it or to forge a passport altogether. Other authorities, for example, the border police in other countries, can verify a passport's authenticity. If they trust the authority that issued the document, the information contained in it is accepted as true.

A digital certificate serves two purposes:

- It establishes the owner's identity.
- It makes the owner's public key available.

Similar to a passport, a certificate must be issued by a trusted authority, a Certification Authority (CA) and, similar to a passport, it is issued only for a limited time. When its expiration date has passed, it must be replaced.

Digital certificates are used for identifying either end of a an SSL connection and contain information required to establish trust.

A digital certificate is a digitally signed data structure that binds a public key to the identity of the private key's owner. The use of digital certificates ensures that the user of a public key can be confident of the ownership of the corresponding private key. If you intend using SSL, you must always configure server authentication.

In this example, we show how the Spiffy Computer Company's requirement for connecting through remote DB2 clients drove it to implement digital certification. We also describe the benefits of digital certification.

11.2.1 Background information for the certificate scenario

The Spiffy Computer Company's CIO has a business need to acquire a customer relationship management (CRM) system. The CIO convinces the CEO that a CRM system can help Spiffy Computer Company improve revenue. The CRM project is approved and a project team is formed. The CRM application server is selected and an external software integration company is procured for web application development.

Because of a lack of office space, the new application server and the application developers are located in rented facilities and connected remotely. During the project, the application server should be connected from outside the company's main server with access restricted by security policy. The security team suggests using the SSL protocol between the new CRM application server and the DB2 for z/OS server. This SSL protocol must be certified by z/OS digital certification.

Figure 11-6 shows how the new CRM project application server is connected to DB2 for z/OS.

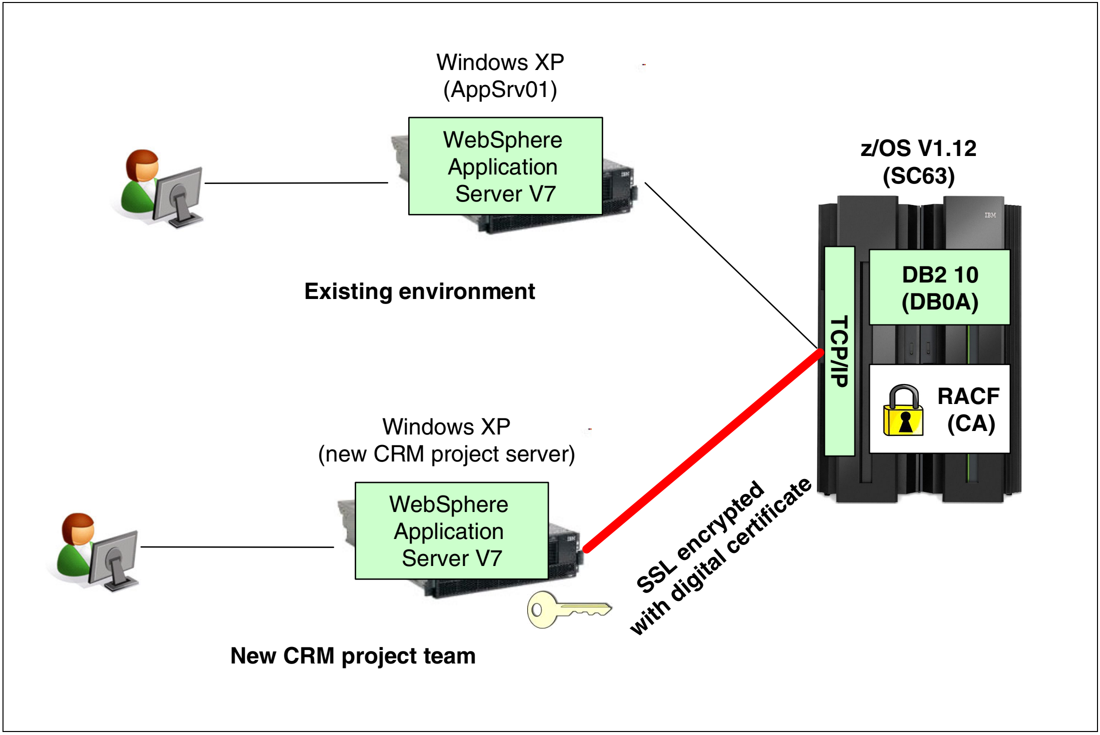


Figure 11-6 SSL with z/OS digital certification connection

11.2.2 Implementation scenario

For this implementation, proper planning is needed. Several specialists from different areas are involved in the setup: RACF and security coordinators, network engineers, DB2 system programmers, a WebSphere Application Server administrator, and application developers.

The implementation procedure for the SSL configuration between the CRM application server and the z/OS database server involves the following steps.

1. Configuring the Application Transparent Transport Layer Security policy agent
2. Defining the AT-TLS policy
3. Configuring a DB2 secure port
4. Creating digital certificates

We detail the steps in the following sections.

Configuring the Application Transparent Transport Layer Security policy agent

We do not go into details about how to configure an Application Transparent Transport Layer Security (AT-TLS) policy agent. A network administrator and RACF administrator configure this policy, but z/OS system programming is also involved.

The agent name is PAGENT and it runs as a UNIX process so it can be started either from the UNIX System Services shell or as a z/OS started task.

For detailed configuration and RACF settings, refer to *DB2 9 for z/OS: Configuring SSL for Secure Client-Server Communications*, REDP-4630.

In our case, we decide to use a z/OS started task. The Spiffy Computer Company applied PAGENT started task JCL is listed in Example 11-1.

Example 11-1 Spiffy Computer Company PAGENT STC JCL

```
//PAGENT PROC
//PAGENT EXEC PGM=PAGENT,REGION=OK,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE. Other environment variables can also be
//* specified via STDENV.
//*
//STDENV DD DSN=DB2R5.TCPPARMS(PAENV),DISP=SHR
//* ============================================================
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively. But
//* normally, PAGENT doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the PAGENT_LOG_FILE environment variable, and defaults to
//* /tmp/pagent.log. When the -d parameter is specified, however,
//* output is also written to stdout.
//* ============================================================
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Environment variables can be either configured in an MVS data set or a z/OS UNIX file specified by the STDENV DD to run with the required configuration. In our example, we have configured the environment variables for PAGENT in the MVS data set DB2R5.TCPPARMS and member PAENV. See the DD in bold in Example 11-1 on page 243 and the contents in Example 11-2.

Example 11-2 PAGENT environment file PAENV

```
TZ=UTC-2
PAGENT_CONFIG_FILE=//'DB2R5.TCPPARMS(PAGENT)'
PAGENT_LOG_FILE=SYSLOGD
```

TZ specifies the local time zone and PAGENT_CONFIG_FILE specifies the PAGENT configuration file to use. PAGENT_LOG_FILE specifies the log file name used by PAGENT. It is set to log policy agent messages to SYSLOGD. The contents are shown later in Example 11-13 on page 255.

We then need to define the following RACF settings:

1. Define the PAGENT-started task to RACF.
2. Define a user ID, PAGENT, for the PAGENT-started task.
3. Associate user ID, PAGENT, with the PAGENT-started task.
4. Permit authorized users' access to start and stop the PAGENT-started task.
5. Restrict unauthorized users access to the UNIX **pasearch** command.

Example 11-3 shows the definitions of the main configuration file, PAGENT, for a single TCP/IP stack environment. The AT-TLS policies are defined in a separate image-specific configuration file, TTLSPOL, for the TCP/IP image.

Example 11-3 PAGENT configuration file

```
# LogLevel statement
# SYSERR, OBJERR, PROTERR, and WARNING messages are logged.
LogLevel 15
#
# TcpImage statement
# TCP/IP image: TCPIP
# Path to image-specific configuration file: SYS1.TCPPARMS(TTLSPOL)
# FLUSH parameter specified to delete existing policy data in the
# stack on PAGENT startup or when the configuration files change.
# PURGE parameter specified to delete active policy data from the
# stack and Policy Agent when PAGENT is shut down normally.
#TcpImage TCPIP /u/a048503/ttlspol.txt FLUSH PURGE
TcpImage TCPIP FLUSH PURGE
TTLSConfig //'DB2R5.TCPPARMS(TTLSPOL)'
# Here is an alternate way to specify a common AT-TLS policy for the
# TCPIP image:
#
# TcpImage TCPIP FLUSH PURGE
# TTLSConfig //'DB2R5.TCPPARMS(TTLSPOL)' FLUSH PURGE
#
# TTLSConfig //'SYS1.TCPPARMS(TTLSPOL)' FLUSH PURGE
```

AT-TLS support is controlled by the TTLS or NOTTLS parameter on the TCPCONFIG statement in TCPIP profile. AT-TLS is enabled by specifying TCPCONFIG TTLS.

Defining the AT-TLS policy

Policy conditions consist of a variety of selection criteria that act as filters for AT-TLS rules. Network administrator and security administrator are involved. Traffic can be filtered based on local addresses, remote addresses, local port range, remote port range, job name, user identification, and direction. An AT-TLS policy is defined with the TTLSRule statement that specifies the set of conditions that are compared against the connection being verified by TCP/IP. The rule conditions are:

LocalAddr	Local IP address or addresses
RemoteAddr	Remote IP address or addresses
LocalPortRange	Local port or ports
RemotePortRange	Remote port or ports
Jobname	Job name of the owning application or wildcard job name
Userid	User ID of the owning process or wildcard user ID
Direction	Inbound if applied to a passive socket (established by accept), Outbound if applied to an active socket (established by connect), or both.

Example 11-4 shows an AT-TLS sample policy. DB0ADIST is the DB2 subsystem DIST address space name and 38362 is the secure port.

Example 11-4 AT-TLS policy for the Spiffy Computer Company

```
TTLSRule DB2ASecureServer
 {
  LocalPortRange 38362
  JobName DB0ADIST
  Direction Inbound
  TTLSGroupActionRef DB0ASecureGrpAct
  TTLSEnvironmentActionRef DB0ASecureEnvAct
 }
 TTLSGroupAction DB0ASecureGrpAct
 {
  TTLSEnabled On
  Trace 15
 }
 TTLSEnvironmentAction DB0ASecureEnvAct
 {
  TTLSKeyRingParms
  {
    Keyring DB2AKEYRING
  }
  HandshakeRole Server
 }
```

You can see the DB2AKEYRING in bold. It is the keyring information needed when you configure the digital certificate environment.

An application designated as HandshakeRole Server must have a user or site certificate on its key ring. It must have access to the private keys of its user or site certificate. It might also need other certificates on its key ring, required for authentication. It is the client authentication setting during the handshake phase. This is the minimum level of security to authenticate the communication between a server and its client: using just server authentication.

Other options are HandshakeRole ServerWithClientAuth or Client.

Configuring a DB2 secure port

To implement SSL support for a DB2 server, you need to make sure that the TCP/IP SQL Listener service task of DDF is capable of listening to a secondary secure port for inbound SSL connections. The DB2 system programmer and the network administrator are involved in this task.

The DDF SQL TCP/IP Listener accepts regular (non-SSL) connections on the DRDA port, whereas the secure port accepts only SSL connections to provide secure communications with a partner that uses the SSL protocol.

To define a secure port to DB2, you can either specify the secure port number during DB2 installation by using the Distributed Data Facility Panel 2 (DSNTIP5) or you can specify the secure port by updating the DDF communication record in the BSDS by using the Change Log Inventory (DSNJU003) utility.

We use DSNJU003 utility to define the DDF secure port, as shown in Example 11-5.

Example 11-5 DB2 secure port define using DSNJU003

```
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB  DD  DISP=SHR,DSN=DB0AT.SDSNLOAD
//SYSUT1   DD  DISP=OLD,DSN=DB0AB.BSDS01
//SYSUT2   DD  DISP=OLD,DSN=DB0AB.BSDS02
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *
 DDF    LOCATION=DB0A,SECPORT=38362
```

Important: If you are using DB2 data sharing, the secure port for each DB2 member of the group should be the same, just as the DRDA PORT for each member should also be the same.

Creating digital certificates

The z/OS Security Server, RACF, supports the management of digital certificates in a z/OS environment. Other security products, such as native key database gskkyman, can also support the management of digital certificates. We use RACF as the security product to manage digital certificates.

You can use RACF to create, register, store, and administer digital certificates and their associated private keys, and build certificate requests that can be sent to a certificate authority for signing. You can also use RACF to manage key rings of stored digital certificates.

Digital certificates and key rings are managed in RACF primarily by using the RACDCERT command. In this section, we describe how the RACF administrator can use the RACDCERT command to administer digital certificates and key rings.

Important: The job name specified was DB0ADIST in the AT-TLS policy (Example 11-4), which indicates to AT-TLS that the DB0ADIST started task is the owning application for this AT-TLS rule. Specifying the job name also implies that the user ID associated with DB0ADIST will be used implicitly by AT-TLS to invoke any necessary RACDCERT commands.

So, the RACF command in Example 11-6 should be issued for the DIST started task owner.

Example 11-6 Required DIST owner authority

```
SETROPTS CLASSACT(DIGTCERT DIGTRING)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(STC) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(STC) ACCESS(READ)
SETR RACLIST (DIGTRING) REFRESH
SETR RACLIST (DIGTCERT) REFRESH
SETR RACLIST (FACILITY) REFRESH
```

The Spiffy Computer Company RACF administrator creates the diagram shown in Figure 11-7 to show all the people involved in the procedure needed to create digital certificate file and apply it to WebSphere applications.

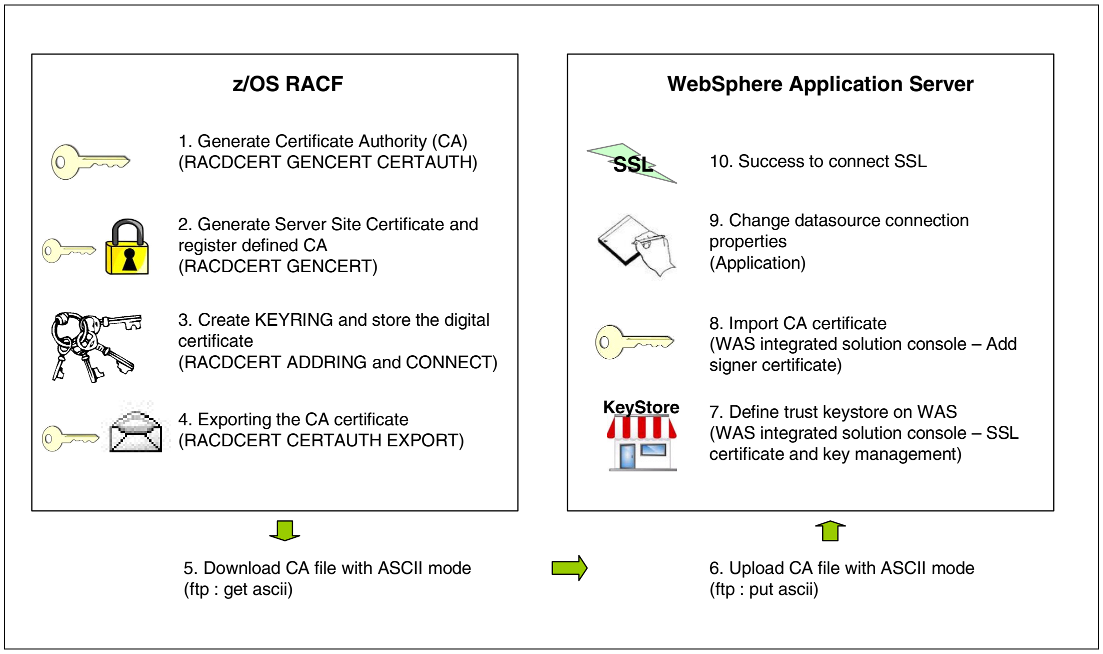

z/OS RACF
1. Generate Certificate Authority (CA) (RACDCERT GENCERT CERTAUTH)
2. Generate Server Site Certificate and register defined CA (RACDCERT GENCERT)
3. Create KEYRING and store the digital certificate (RACDCERT ADDRING and CONNECT)
4. Exporting the CA certificate (RACDCERT CERTAUTH EXPORT)
WebSphere Application Server
SSL
10. Success to connect SSL
9. Change datasource connection properties (Application)
8. Import CA certificate (WAS integrated solution console – Add signer certificate)
KeyStore
7. Define trust keystore on WAS (WAS integrated solution console – SSL certificate and key management)
5. Download CA file with ASCII mode (ftp : get ascii)
6. Upload CA file with ASCII mode (ftp : put ascii)

Figure 11-7 Create procedure digital certificate and import into WebSphere Application Server

According to this procedure, the Spiffy Computer Company RACF administrator creates and defines the Certificate Authority (CA) environment to be used by the WebSphere Application Server administrator:

1. Generate Certificate Authority.

   To generate a CA in RACF, you need to use the RACDCERT GENCERT command with the CERTAUTH option. You can use ID (certificate-owner) or SITE or CERTAUTH. ID means a user certificate associated with the specified user ID, SITE means a site certificate and CERTAUTH means certificate-authority certificate. If you do not specify anything, the default is the ID that is issuing the command. Use CERTAUTH to create the certificate-authority certificate, as shown in Example 11-7. Label names are case sensitive.

   Example 11-7 Create certificate-authority certificate

   ```
   RACDCERT CERTAUTH GENCERT -
            SUBJECTSDN(OU('Jimbo Server CA')
   ```

```
                  O('IBM') -
                  L('MMAA') -
                  SP('SEOUL') -
                  C('KOREA')) -
        NOTAFTER(DATE(2030-12-31)) -
        WITHLABEL('Jimbo Server CA') -
        KEYUSAGE(CERTSIGN)
```

Tip: When creating certificates for the NSS server's key ring, avoid using lowercase alphabetic characters, blanks, and the characters *, %, and & in the certificate's label.

For details, refer to *z/OS V1R12.0 Security Server RACF Command Language Reference*, SA22-7687.

We can use our information in the SUBJECTSDN field. It conforms with the X.509 standard, which is widely used standard for defining digital certificates. The X.509 digital certificate is a data structure that contains, at minimum, the following fields:

CommonName	Specified with the CN subkeyword.
Title	Specified with the Tsubkeyword.
Organizational Unit	Specified with the OU subkeyword. Multiple values can be specified for the organizational unit.
Organization	Specified with the O subkeyword.
Locality	Specified with the L subkeyword
State/Province	Specified with the SP subkeyword.
Country	Specified with the C subkeyword.

This information is also used to generate the Server Site Certificate in the following step.

2. Generate the Server Site Certificate and register the defined CA.

   To generate a site certificate signed by the CA certificate that we just created in RACF, use the RACDCERT command. This step creates the site certificate and also specifies the certificate in the SIGNWITH option.

   The SIGNWITH option specifies the certificate with a private key that is signing the certificate. If it is not specified, the default is to sign the certificate with the private key of the certificate that is being generated, which creates a self-signed certificate. The signing certificate must belong to the user ID executing the command, or SITE or CERTAUTH. If the SITE and CERTAUTH keywords are omitted, the signing certificate owner defaults to the user ID of the command issuer. We need to use CERTAUTH keywords and its label, as shown in Example 11-8.

Example 11-8 Generate Server Site Certificate

```
RACDCERT ID(STC) GENCERT -
        SUBJECTSDN(CN('Jimbo Server Certificate') -
                  OU('SVL') -
                  O('SVLSEC') -
                  C('USA'))
        NOTAFTER(DATE(2030-12-31))
        WITHLABEL('Jimbo Server Certificate')
        SIGNWITH(CERTAUTH LABEL('Jimbo Server CA'))
```

The owner’s ID STC is the DIST address space’s (DB0ADIST) owner.

3. Create the KEYRING and store the digital certificate.

   Now we need to create the KEYRING to hold keys. This KEYRING was defined when we defined the AT-TLS policy in Example 11-4 on page 245.

   Using the commands in Example 11-9, we create the KEYRING and store the digital certificate. This is the final step to create the digital certificate before exporting it.

   Example 11-9 Create the KEYRING and store the digital certificate

```
RACDCERT ID(STC) ADDRING(DB2AKEYRING)

RACDCERT ID(STC)CONNECT(CERTAUTHLABEL('JimboServerCA')RING(DB2AKEYRING))

RACDCERT ID(STC) CONNECT(ID(STC) -
                 LABEL('Jimbo Server Certificate') -
                 RING(DB2AKEYRING) DEFAULT)
```

4. Export the CA certificate.

   After this step, we can see the CA certificate and move it to another system or person, as shown in Example 11-10.

   Example 11-10 Export the CA certificate

```
RACDCERT CERTAUTH EXPORT(LABEL('Jimbo Server CA')) -
                          DSN('DB2R7.JIMBO2.CACERT')
```

   Before going to the next procedure, if you want to delete the already defined CA certificates or redefine it, you need to delete it using the RACDCERT command, as shown in Example 11-11.

   Example 11-11 CA certificates delete command

```
RACDCERT DELETE (LABEL('Jimbo Server CA')) CERTAUTH
RACDCERT DELETE (LABEL('Jimbo Server Certificate')) ID(STC)
RACDCERT DELRING (DB2AKEYRING) ID(STC)
```

5. Download the CA certificate file through FTP.
6. Upload the CA certificate file through FTP.

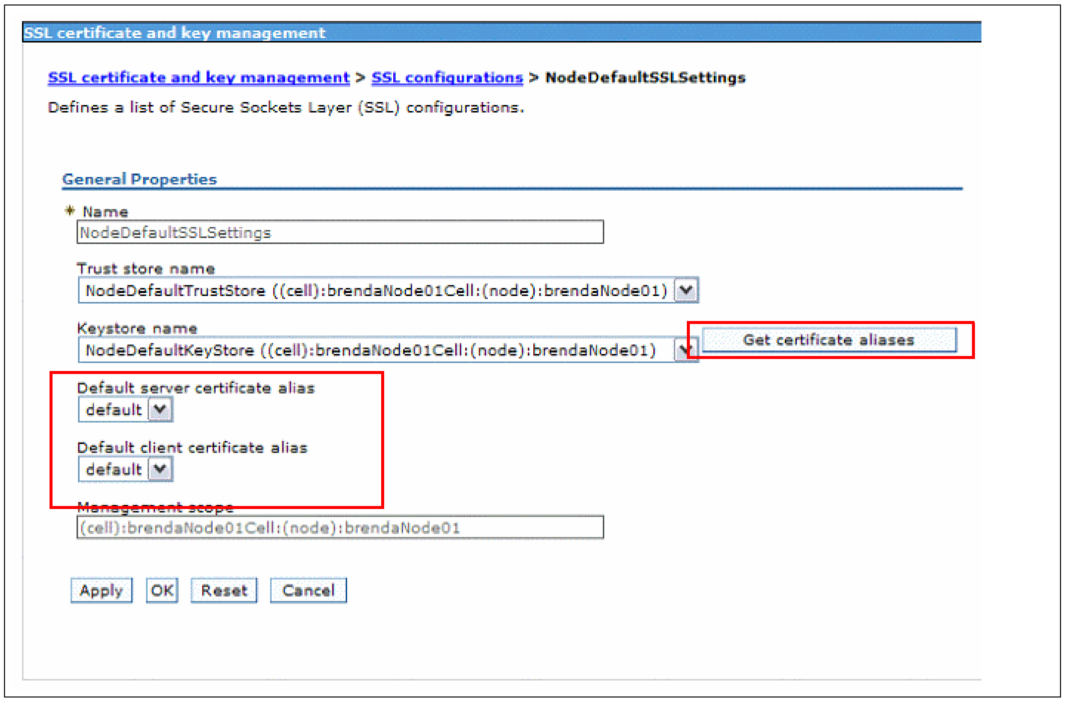
   We can download and update the z/OS RACF created CA file and upload it.

   Important: The CA file that is created by z/OS RACF should be downloaded through FTP in ASCII mode, *not* in BINARY mode.

7. Define a trust keystore in the IBM WebSphere Integrated Solutions Console.

   After uploading the CA certificate on the WebSphere platform, the WebSphere administrator needs to define a trust keystore in the WebSphere Integrated Solutions Console.

To define the trust keystore, select **SSL certificate and key management** → **SSL configuration** → **NodeDefaultSSLSettings**. The window shown in Figure 11-8 opens.


SSL certificate and key management
SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings
General Properties
Get certificate aliases
Apply
OK
Reset
Cancel

Defines a list of Secure Sockets Layer (SSL) configurations.

* Name: NodeDefaultSSLSettings

Trust store name: NodeDefaultTrustStore ((cell):brendaNode01Cell:(node):brendaNode01)

Keystore name: NodeDefaultKeyStore ((cell):brendaNode01Cell:(node):brendaNode01)

Default server certificate alias: default

Default client certificate alias: default

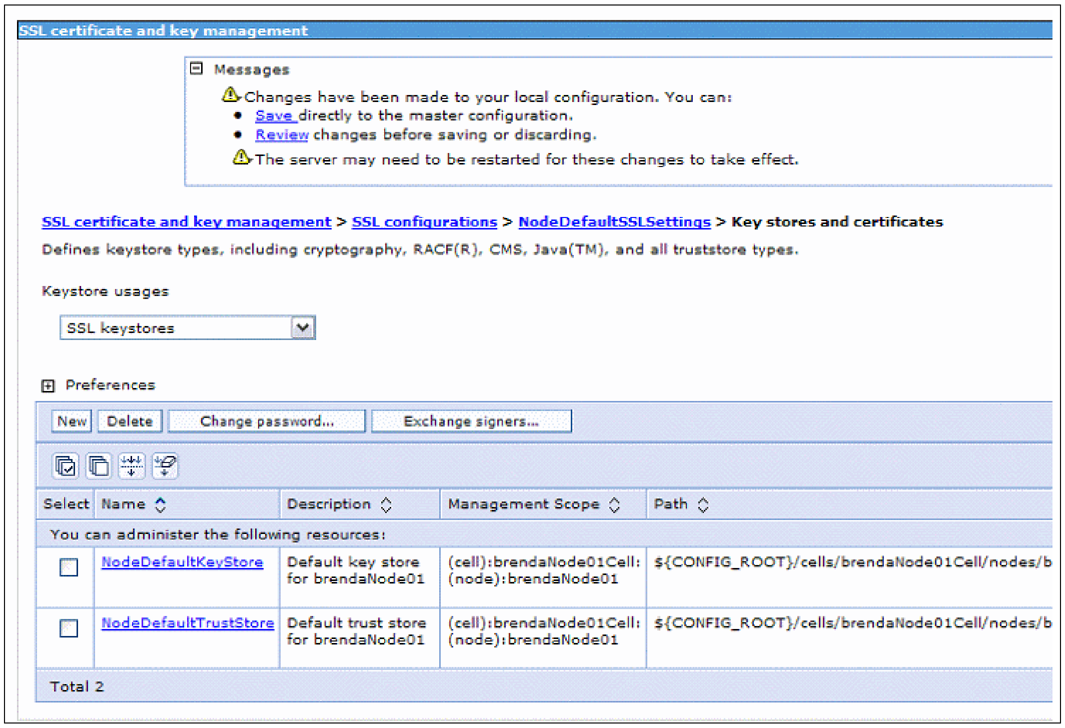
Management scope: (cell):brendaNode01Cell:(node):brendaNode01

Figure 11-8 Node default SSL settings

When you click **Get certificate aliases**, you can see the Default server certificate alias and the Default client certificate alias drop-down menus change from none to default.

You can check the default trust keystore that was created.

Figure 11-9 shows the NodeDefaultTrustStore that we use for the SSL connection.



SSL certificate and key management

Messages

Changes have been made to your local configuration. You can:
- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Key stores and certificates

Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all truststore types.

Keystore usages

SSL keystores

Preferences

New | Delete | Change password... | Exchange signers...

Select	Name	Description	Management Scope	Path
You can administer the following resources:				
☐	NodeDefaultKeyStore	Default key store for brendaNode01	(cell):brendaNode01Cell:(node):brendaNode01	${CONFIG_ROOT}/cells/brendaNode01Cell/nodes/b
☐	NodeDefaultTrustStore	Default trust store for brendaNode01	(cell):brendaNode01Cell:(node):brendaNode01	${CONFIG_ROOT}/cells/brendaNode01Cell/nodes/b

Total 2

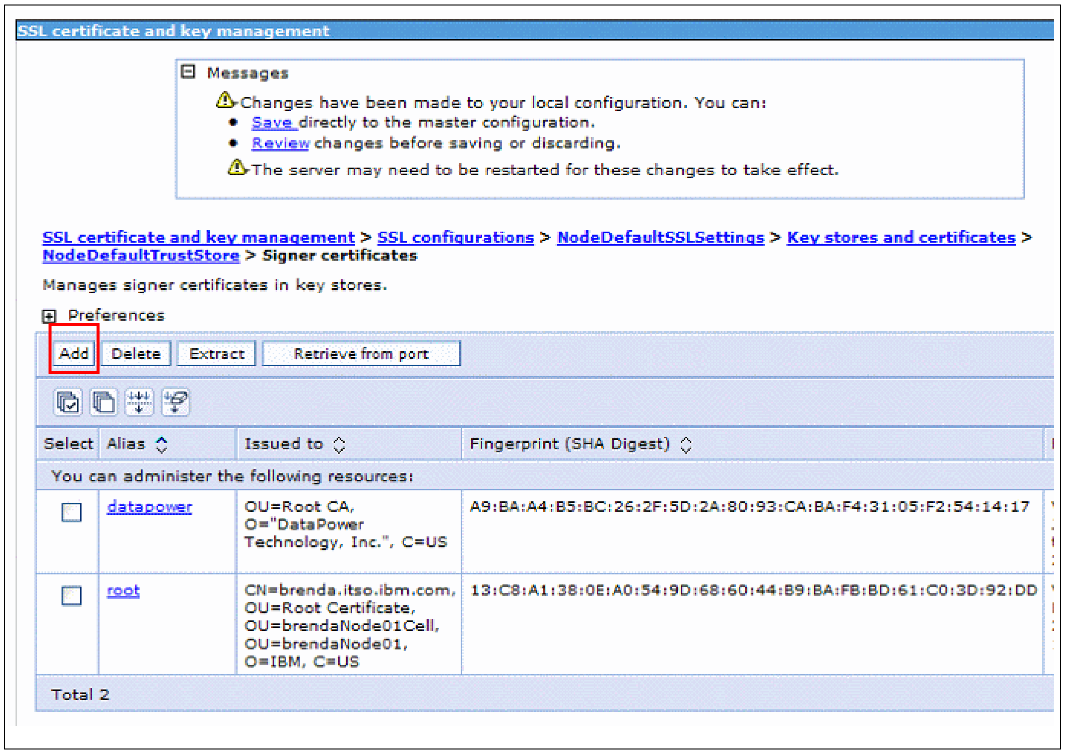
Figure 11-9 Node default keystore creation check

8. Import the CA certificate.

   After creating the node default keystore, we need to import the z/OS RACF authorized CA certificate into the application server.

Click **NodeDefaultTrustStore**. You can now see and click **Signer Certificates**, on the right of the console in Figure 11-10.

You can see two already defined resources. We import the z/OS RACF authorized CA certificate form this console window.


SSL certificate and key management

Messages

Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Key stores and certificates > NodeDefaultTrustStore > Signer certificates

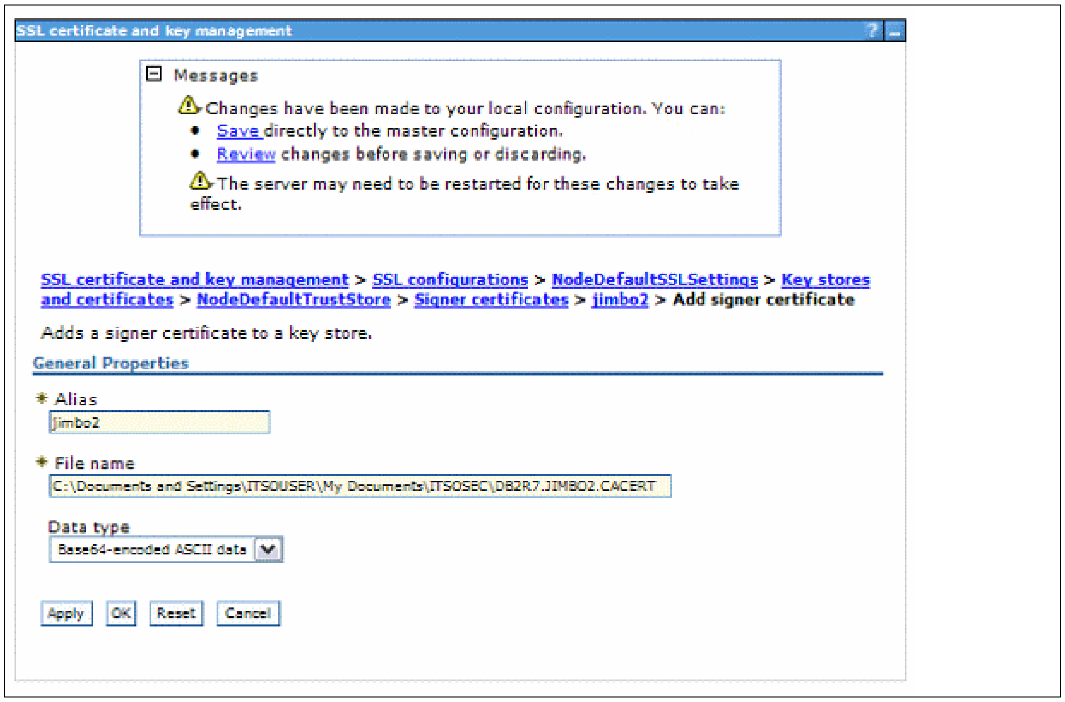
Manages signer certificates in key stores.

Select	Alias	Issued to	Fingerprint (SHA Digest)
You can administer the following resources:			
☐	datapower	OU=Root CA, O="DataPower Technology, Inc.", C=US	A9:BA:A4:B5:BC:26:2F:5D:2A:80:93:CA:BA:F4:31:05:F2:54:14:17
☐	root	CN=brenda.itso.ibm.com, OU=Root Certificate, OU=brendaNode01Cell, OU=brendaNode01, O=IBM, C=US	13:C8:A1:38:0E:A0:54:9D:68:60:44:B9:BA:FB:BD:61:C0:3D:92:DD

Total 2

Figure 11-10 Signer certificates management console

To add the signer certificates, you need to check the right file path for the z/OS RACF created CA certificate. The data type is ASCII. Refer to Figure 11-11.



Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Key stores and certificates > NodeDefaultTrustStore > Signer certificates > jimbo2 > Add signer certificate

Adds a signer certificate to a key store.

General Properties

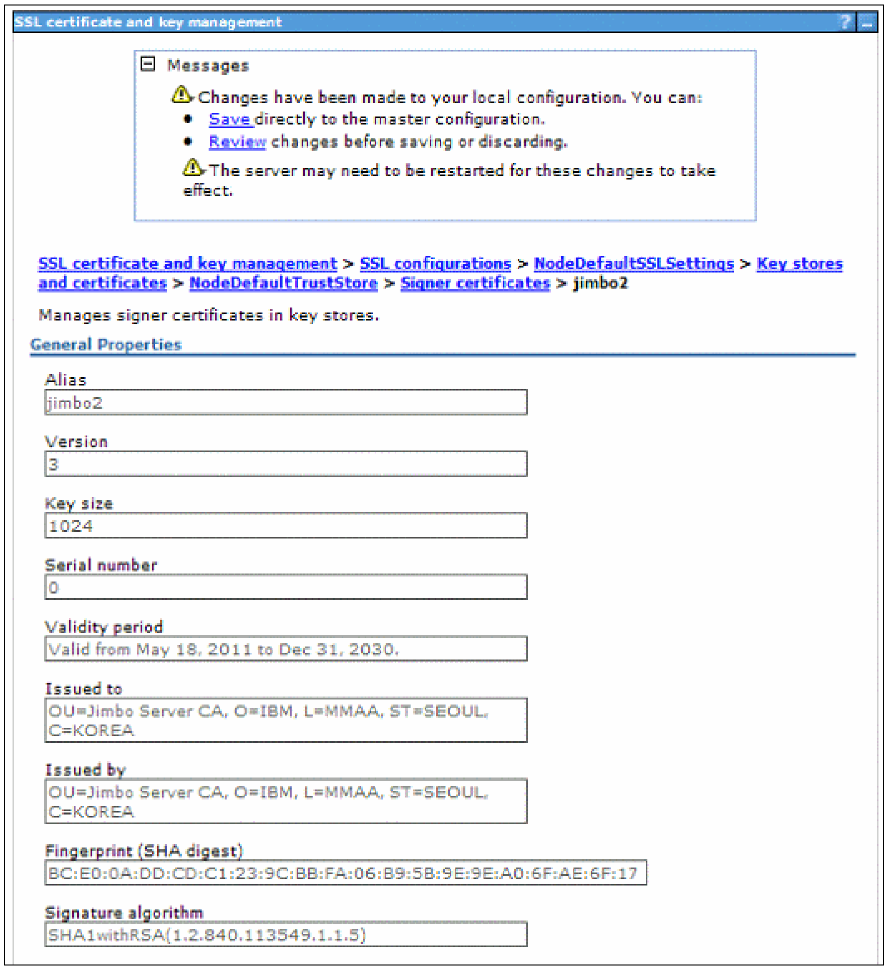
* Alias: jimbo2

* File name: C:\Documents and Settings\ITSOUSER\My Documents\ITSOSEC\DB2R7.JIMBO2.CACERT

Data type: Base64-encoded ASCII data

Figure 11-11 Add signer certificates

If you have specified the right CA certificate and the right file path, you can see the CA certificate information after adding the signer certificates, as shown in Figure 11-12.



Messages

Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Key stores and certificates > NodeDefaultTrustStore > Signer certificates > jimbo2

Manages signer certificates in key stores.

General Properties

Alias: jimbo2

Version: 3

Key size: 1024

Serial number: 0

Validity period: Valid from May 18, 2011 to Dec 31, 2030.

Issued to: OU=Jimbo Server CA, O=IBM, L=MMAA, ST=SEOUL, C=KOREA

Issued by: OU=Jimbo Server CA, O=IBM, L=MMAA, ST=SEOUL, C=KOREA

Fingerprint (SHA digest): BC:E0:0A:DD:CD:C1:23:9C:BB:FA:06:B9:5B:9E:9E:A0:6F:AE:6F:17

Signature algorithm: SHA1withRSA(1.2.840.113549.1.1.5)

Figure 11-12 Signer certificate's certificates information

After saving this configuration, all the configurations will connect to z/OS data server with the CA certificate that was issued by z/OS RACF.

9. Change the data source connection properties

   To use the SSL connection, the application's database connection information should be changed. We need to change the port number that was defined for the DB2 secure port in Example 11-5 on page 246 and also change the connection method setting.

   From the application's point of view, enabling SSL in IBM Data Server Driver for JDBC and SQLJ requires only one change: the property sslConnection must be set to true. This forces the Data Server Driver for JDBC and SQLJ to use an SSL socket to connect to DB2 database servers. This property can be set through both DB2DataSource and DriverManager interfaces. Example 11-12 shows how to do so with the DB2DataSource.

Example 11-12 Application change to activate SSL protocol

```
DB2ConnectionPoolDataSource dataSource = new DB2ConnectionPoolDataSource();
dataSource.setDatabaseName(databaseName);
dataSource.setServerName(serverName);
dataSource.setDriverType(Integer.valueOf(driverType));
dataSource.setPortNumber(Integer.valueOf(portNumber));
dataSource.setSslConnection(true);
```

This example does not use the WebSphere Application Server supplied data source, so we need to change the application source directly. If you use the WebSphere Application Server supplied data source, you can also change it in the WebSphere Integrated Solutions Console.

Now, Spiffy Computer Company connects using the SSL protocol between the new CRM application server and DB2 for z/OS database server.

10. Run the application and check the SSL connection.

    After running the application, you can see the text SSL message reported in Example 11-13 in the SYSLOGD data set that we defined in Example 11-2 on page 244, the AT-TLS policy agent environment file.

Example 11-13 SSL communication messages from SYSLOGD

```
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 19 22:02:52 wtsc63/STC      TCPIP    TTLSÝ262350¨: 18:02:52 TCPIP
   EZD1281I TTLS Map   CONNID: 00001B34 LOCAL: 9.12.6.70..38362
REMOTE: 9.30.28.119..1680 JOBNAME: DBOADIST USERID: STC TYPE: InBound
STATUS: Enabled RULE: DB2ASecureServer ACTIONS: DBOASecureGrpAct
DBOASecureEnvAct **N/A**
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 19 22:02:52 wtsc63/STC      TCPIP    TTLSÝ262350¨: 18:02:52 TCPIP
   EZD1283I TTLS Event GRPID: 00000005 ENVID: 00000000 CONNID:
00001B34  RC:    0 Connection Init
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 19 22:02:52 wtsc63/STC      TCPIP    TTLSÝ262350¨: 18:02:52 TCPIP
   EZD1282I TTLS Start GRPID: 00000005 ENVID: 0000000F CONNID:
00001B34 Initial Handshake ACTIONS: DBOASecureGrpAct DBOASecureEnvAct
**N/A** HS-Server
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 19 22:02:52 wtsc63/STC      TCPIP    TTLSÝ262350¨: 18:02:52 TCPIP
   EZD1283I TTLS Event GRPID: 00000005 ENVID: 0000000F CONNID:
00001B34  RC:    0 Initial Handshake 7EC4DA18 7EDB9358 TLSV1    05
```

If you see a `EZD1283I` message with non-zero return code, such as `RC: 5006`, during the initial handshake, you need to check the CA certificate generation procedure.

The return code information can be checked in “AT-TLS return codes” in *z/OS Communications Server: IP Diagnosis Guide,* GC31-8782.

11.2.3 Advanced implementation scenario

The previous scenario was about CA *server authentication only* during the handshake phase. As mentioned, the Secure Sockets Layer (SSL) protocol also supports client authentication during the handshake phase.

The SSL protocol provides server authentication as the minimum level of security. It uses the Server Authentication mechanism to secure communications between a server and its client and allows the client to validate the authenticity of the server.

The SSL protocol provides client authentication as an additional level of authentication and access control. It enables a server to validate the certificates of a client at the server and thus prevents the client from obtaining a secure connection without an installation-approved certificate.

Client authentication is optional and, if used, can provide the following three levels of authentication:

- Level 1 authentication is performed by system SSL. A client passes a digital certificate to a server as part of the SSL handshake. To successfully pass the required authentication, the Certificate Authority (CA) that signs the client certificate must be trusted by the server, that is, the certificate for the CA must be in the key ring that the server uses and designates as trusted.
- Level 2 (addition to level 1) authentication requires that a client certificate be registered with RACF (or other SAF-compliant security products) and mapped to a valid user ID. When AT-TLS receives the client certificate during the SSL handshake, it queries RACF to verify that the certificate maps to a valid user ID before allowing a secure connection to be established. This level of client authentication provides additional access control at the server and ensures that the client is known to have a valid user ID on the server host.
- Level 3 (addition to levels 1 and 2) authentication provides the capability to restrict access to a server based on the user ID associated with a client certificate. A client can access a server only if the client itself is valid to the server, its certificate is valid, and a user ID associated with the certificate is valid. This level of authentication uses the RACF SERVAUTH general resource class to restrict access to the server based on the user ID of the client. If the SERVAUTH general resource class is not active or the SERVAUTH profile for the server is not defined, AT-TLS assumes that this level of authentication is not requested. However, if the SERVAUTH general resource class is active and the server's SERVAUTH profile is defined, a remote secure connection is be established only if the user ID that is associated with the client certificate is permitted to the server's SERVAUTH profile. Otherwise, the secure connection is not established and the connection itself is dropped.

The advanced Spiffy Computer Company's scenario uses *server and client authentication*. For the Level 2 client authentication, we need to change AT-TLS policy and register the CA certificate to RACF. Figure 11-13 shows the difference between the basic scenario and the advanced scenario.

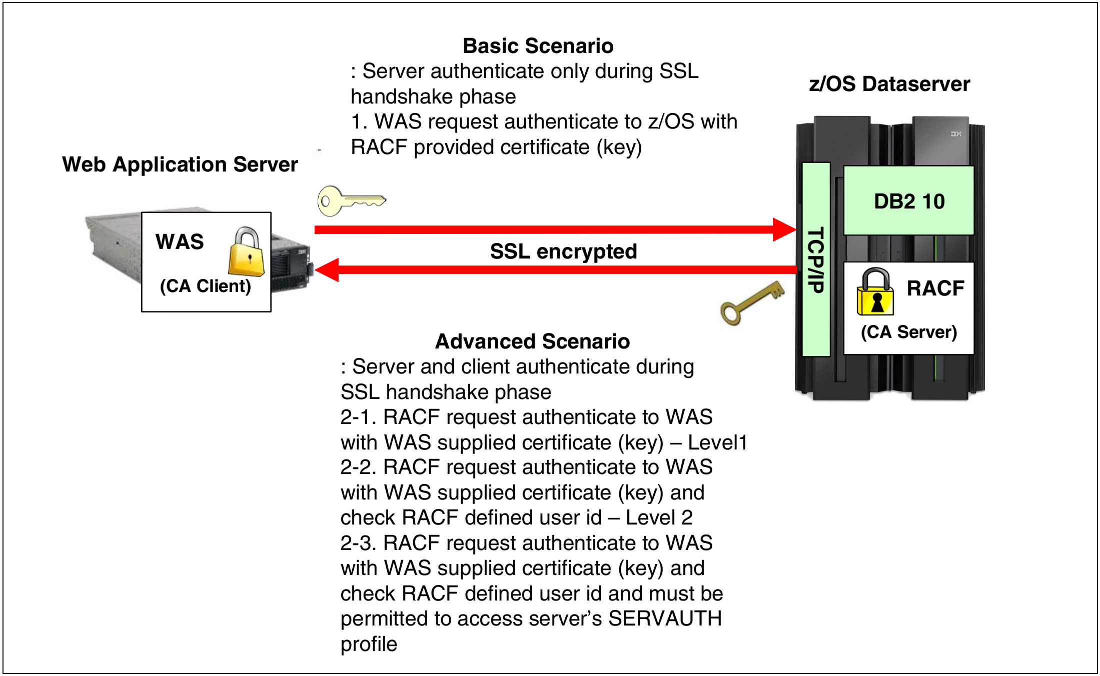
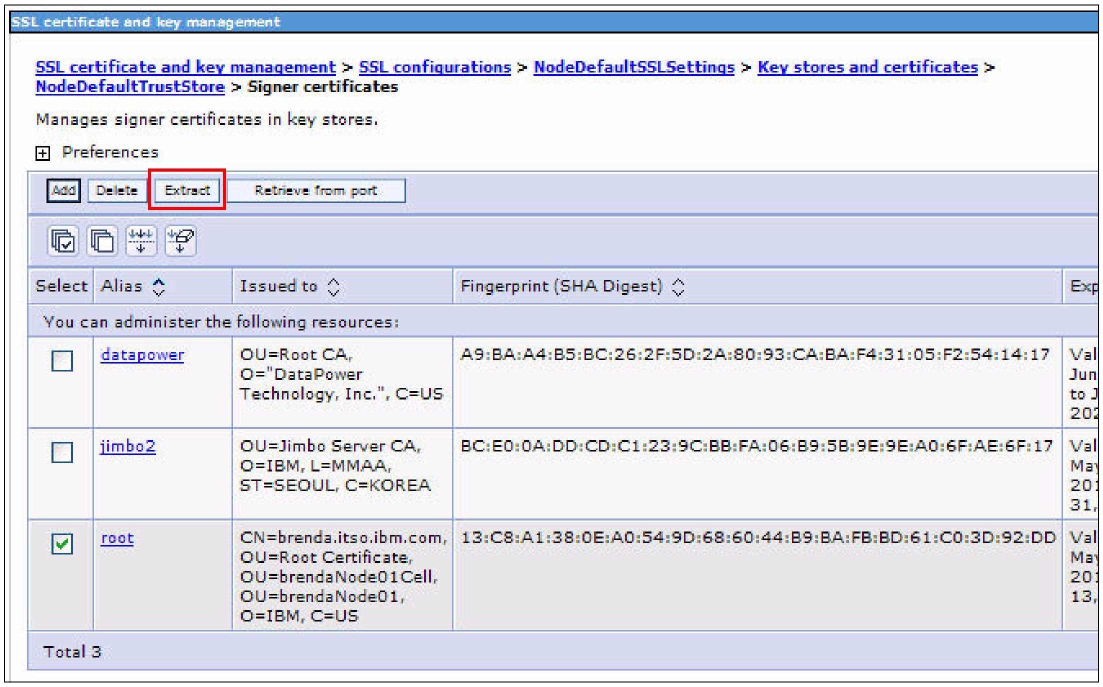

Basic Scenario
: Server authenticate only during SSL handshake phase
1. WAS request authenticate to z/OS with RACF provided certificate (key)
z/OS Dataserver
Web Application Server
WAS
(CA Client)
SSL encrypted
TCP/IP
DB2 10
RACF
(CA Server)
Advanced Scenario
: Server and client authenticate during SSL handshake phase
2-1. RACF request authenticate to WAS with WAS supplied certificate (key) – Level1
2-2. RACF request authenticate to WAS with WAS supplied certificate (key) and check RACF defined user id – Level 2
2-3. RACF request authenticate to WAS with WAS supplied certificate (key) and check RACF defined user id and must be permitted to access server's SERVAUTH profile

Figure 11-13 Server and client authentication during the SSL handshake

Spiffy Computer Company now implements SSL authentication Level 2. First, the client certificate needs to be registered to the RACF trusted key. To obtain the WebSphere Application Server client certificate, select **SSL certificate and key management** → **SSL configurations** → **NodeDefaultSSLSettings** → **Key stores and certificates** → **NodeDefaultTrustStore** → **Signer certificates** in the Integrated Solutions Console, as shown in Figure 11-14.


SSL certificate and key management
SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Key stores and certificates > NodeDefaultTrustStore > Signer certificates
Manages signer certificates in key stores.
Preferences
Add
Delete
Extract
Retrieve from port

Select	Alias	Issued to	Fingerprint (SHA Digest)	Exp
You can administer the following resources:				
☐	datapower	OU=Root CA, O="DataPower Technology, Inc.", C=US	A9:BA:A4:B5:BC:26:2F:5D:2A:80:93:CA:BA:F4:31:05:F2:54:14:17	Val Jun to J 202
☐	jimbo2	OU=Jimbo Server CA, O=IBM, L=MMAA, ST=SEOUL, C=KOREA	BC:E0:0A:DD:CD:C1:23:9C:BB:FA:06:B9:5B:9E:9E:A0:6F:AE:6F:17	Val Ma 20 31,
✓	root	CN=brenda.itso.ibm.com, OU=Root Certificate, OU=brendaNode01Cell, OU=brendaNode01, O=IBM, C=US	13:C8:A1:38:0E:A0:54:9D:68:60:44:B9:BA:FB:BD:61:C0:3D:92:DD	Val Ma 20 13,
Total 3				

Figure 11-14 Client (WebSphere Application Server) certificate extract

We need to extract the root certificate from the data set and upload it a z/OS data set in ASCII format.

Now, to have RACF recognize the client's certificate as trusted, we need to register the client certificate to RACF as trusted and connect to the keyring defined in the AT-TLS policy, as shown in Example 11-14.

When you add a trusted certificate, you need to use the RACF defined user ID. In this case, we need to use a DB2 trusted context authid. This user ID is checked for SSL Level 2 authentication. When you connect the certificate to the keyring, the owner should be the keyring's owner ID, which is the DDF started task's owner.

Example 11-14 Registering a client certificate to RACF

```
RACDCERT ID(DB2R7) ADD('DB2R7.JIMBO1.CLIENT.CACERT3') -
        WITHLABEL('WAS CERT') TRUST

IRRD175I The new profile for DIGTCERT will not be in effect until a SETROPTS RE
FRESH has been issued.
 IRRD113I The certificate that you are adding is self-signed.  The certificate i
s added with TRUST status.
 ***

RACDCERT ID(STC) CONNECT(ID(DB2R7) LABEL('WAS CERT') -
        RING(DB2AKEYRING) USAGE(PERSONAL)
```

```
RACDCERT ID(DB2R7) LIST

Digital certificate information for user DB2R7:

  Label: WAS CERT
  Certificate ID: 2QXEwvLZ9+bB4kDDxdnj
  Status: TRUST
  Start Date: 2011/05/18 00:36:02
  End Date:   2026/05/14 00:36:02
  Serial Number:
       >18DB43B515AF<
  Issuer's Name:
       >CN=brenda.itso.ibm.com.OU=Root Certificate.OU=brendaNode01Cell.OU=bre<
       >ndaNode01.O=IBM.C=US<
  Subject's Name:
       >CN=brenda.itso.ibm.com.OU=Root Certificate.OU=brendaNode01Cell.OU=bre<
       >ndaNode01.O=IBM.C=US<
  Subject's AltNames:
    EMail: ProfileUUID:AppSrv01-BASE-78e789cd-103b-4812-8c64-dea448a8f606
  Private Key Type: None
  Ring Associations:
    Ring Owner: STC
    Ring:
       >DB2AKEYRING<

SETROPTS RACLIST(DIGTCERT, DIGTRING) REFRESH
```

Before testing the SSL authentication, we need to change the AT-TLS policy to SSL client authentication Level 2. Example 11-15 shows what we need to change.

Example 11-15 AT-TLS policy change to SSL Level 2 authentication

```
TTLSRule DB0ASecureServer
 {
  LocalPortRange 38362
  JobName DB0ADIST
  Direction Inbound
  TTLSGroupActionRef DB0ASecureGrpAct
  TTLSEnvironmentActionRef DB0ASecureEnvAct
 }
 TTLSGroupAction DB0ASecureGrpAct
 {
  TTLSEnabled On
  Trace 15
 }
 TTLSEnvironmentAction DB0ASecureEnvAct
 {
  TTLSKeyRingParms
  {
    Keyring DB2AKEYRING
  }
  HandShakeRole ServerWithClientAuth
  TTLSEnvironmentAdvancedParms
  {
    ClientAuthType SAFCheck
  }
 }
```

Now we need to refresh the AT-TLS policy with the “F PAGENT,REFRESH” command.

Spiffy Computer Company’s DB2 is now ready to accept secure connections from remote clients that use SSL client and server authentication.

After running the application, we can collect the SYSLOG shown in Example 11-16.

Example 11-16 SYSLOG for SSL Level 2 authentication

```
IEA989I SLIP TRAP ID=X422 MATCHED.  JOBNAME=BPXOINIT, ASID=0034.
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1281I TTLS Map   CONNID: 00002C70 LOCAL: 9.12.6.70..38362
REMOTE: 9.30.28.119..3252 JOBNAME: DBOADIST USERID: STC TYPE: InBound
STATUS: Enabled RULE: DBOASecureServer ACTIONS: DBOASecureGrpAct
DBOASecureEnvAct **N/A**
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1283I TTLS Event GRPID: 00000002 ENVID: 00000000 CONNID:
00002C70  RC:    0 Connection Init
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1282I TTLS Start GRPID: 00000002 ENVID: 00000005 CONNID:
00000000 Environment Create ACTIONS: DBOASecureGrpAct DBOASecureEnvAct
 **N/A**
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1283I TTLS Event GRPID: 00000002 ENVID: 00000006 CONNID:
00000000  RC:    0 Environment Master Create 00000005
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1283I TTLS Event GRPID: 00000002 ENVID: 00000006 CONNID:
00000000  RC:    0 Environment Master Init 7EB2E718
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1283I TTLS Event GRPID: 00000002 ENVID: 00000005 CONNID:
00000000  RC:    0 Environment Link 7EB2E718 00000006
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1282I TTLS Start GRPID: 00000002 ENVID: 00000005 CONNID:
00002C70 Initial Handshake ACTIONS: DBOASecureGrpAct DBOASecureEnvAct
**N/A** HS-ServerWithClientAuth
BPXF060I LOGGED BY SYSLOGD FROM A LOCAL SOURCE     000
May 24 18:12:37 wtsc63/STC      TCPIP    TTLSÝ33816735¨: 14:12:37
TCPIP    EZD1283I TTLS Event GRPID: 00000002 ENVID: 00000005 CONNID:
00002C70  RC:    0 Initial Handshake 7EC4C218 7EB2E718 TLSV1    05
IEA989I SLIP TRAP ID=X422 MATCHED.  JOBNAME=BPXOINIT, ASID=0034.
```

The log confirms that Spiffy Computer Company has SSL Level 2 authentication between WebSphere Application Server and DB2 for z/OS.

11.2.4 Lessons learned

The Internet has created many new global business opportunities for enterprises conducting online services. However, the many security risks associated with conducting online services have resulted in security becoming a major factor for online success or failure.

SSL is the standard security technology for creating an encrypted link between a web server and a browser. Spiffy Computer Company thinks it is also needed between the web server and the data server. This link ensures that all data passed between environments remain private and protected. SSL is an industry standard used for the protection of online transactions with their customers.

DB2 supports SSL protocol by using the z/OS Communications Server IP Application Transparent Transport Layer Security (AT-TLS).

The z/OS Communications Server for TCP/IP (beginning in V1R7 of z/OS) supports the AT-TLS function in the TCP/IP stack for applications that require secure TCP/IP connections. AT-TLS performs TLS on behalf of the application, such as DB2, by invoking the z/OS system SSL in the TCP layer of the TCP/IP stack. The z/OS system SSL supports TLS V1.0, SSL V3.0, and SSL V2.0 protocols. AT-TLS also uses policies that provide system SSL configurations for connections that use AT-TLS. An application continues to send and receive clear text data over the socket while the transmission is protected by the system SSL.

From DB2 10, we can take advantage of SSL client and server authorization with digital certificate. z/OS V1.12 has introduced performance enhancements when using SSL.

11.3 Identity propagation for a remote client application

As we mentioned, remote client access to the enterprise business system is a must, due to IT trends and business requirements. We should have procedure in ready to verify the access and monitor remote connections.

This Spiffy Computer Company needs to support transactions that run on a DB2 subsystem originated outside of z/OS from the Internet and are initiated by users who authenticate their identities on web-based, or distributed, application servers. When a distributed application server establishes a connection to a DB2 subsystem, the connection might be associated with the identity of the distributed application user, as defined in a user registry where the transaction originated, or it might be associated with a shared RACF user ID that was assigned by the z/OS subsystem.

To be effective, customers that audit user activities on DB2 subsystems need both the RACF user ID that is associated with the connection and the user identity that was presented when the user originally accessed the distributed application server.

We already designed and implemented the support for two types of accesses coming from outside the z/OS system for the Spiffy Computer Company: one is initiated by users with DB2 for LUW client and the other is initiated by applications from the WebSphere Application Server. Now we want to identify these workloads and audit them.

11.3.1 Background information

Let us discuss how identity propagation works with DB2 row permissions.

Spiffy Computer Company's audit administrator is worried about system auditing. The workloads coming from outside z/OS, especially web application servers and web applications, keep increasing and their importance is growing, but users are ambiguous, hard to monitor, and audit.

The audit administrator, system security administrator, and DB security administrator want to audit the remote distributed workload. They decide to adopt z/OS security server's identity propagation, which was introduced in z/OS V1R11 and is supported by DB2 10 through a trusted context API.

The plan is that all web application servers should be connecting to the z/OS database server through the DB2 trusted context and network protocol SSL with a CA certificate between the z/OS server and the WebSphere Application Server client. Then the remote user's identification, such as the name, serial number, and so on, will be combined with the previous SQL ID and create a new SQL ID through z/OS identity propagation and the DB2 provided API. SMF will be able to write this information and enable DB2 audit reporting.

Figure 11-15 shows the Spiffy Computer Company's remote client application's authentication and access plan for security.

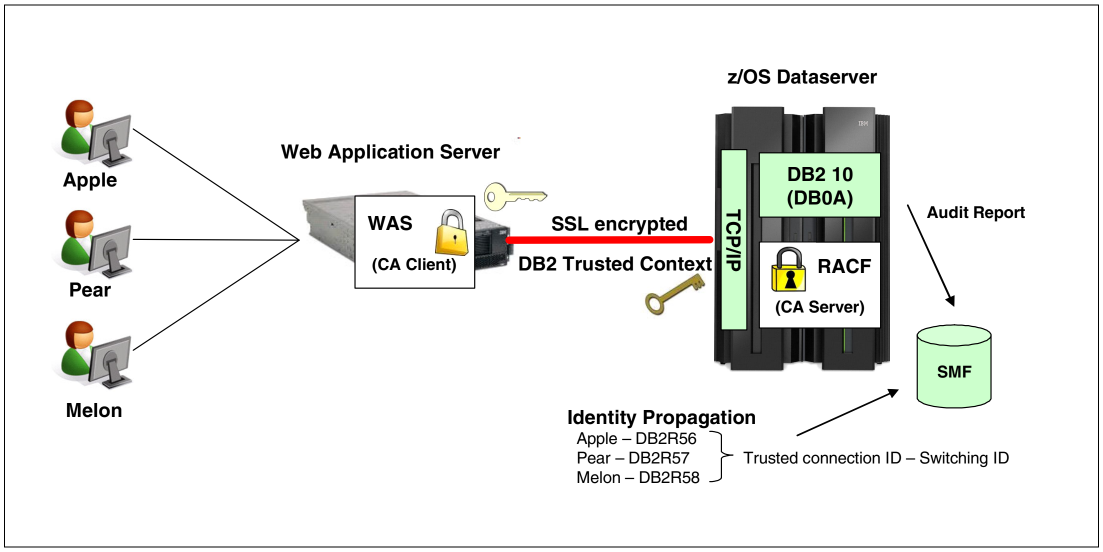


Figure 11-15 Spiffy Computer Company's authentication, data encryption, and auditing plan

Spiffy Computer Company has already encrypted their data with Secure Sockets Layer, which supports server and client authentication during the handshake phase.

Spiffy Computer Company's security and audit environment implements identity propagation. In this scenario, Spiffy Computer Company requires that the employee table should allow employees to see only their own information from the remote accessing web application using the employee's first name as identification.

Example 11-17 shows that the contents of the employee table can be read currently by all people with the table SELECT privilege.

Example 11-17 Spiffy Computer Company's employee table without restrictions

```
SELECT * FROM SPF.EMP ;

EMPNO    FIRSTNME  MIDINIT  LASTNAME  WORKDEPT
=====    ========  =======  ========  ========
PAOLOR8  CHISTINE  I        HAAS      A00
PAOLOR1  ERNIE     L        THOMPSON  B01
PAOLOR6  EVA       D        PULASKI   D21
PAOLOR3  EILEEN    W        HENDERSON E11
PAOLOR5  SEAN               O'CONNELL A00
PAOLOR7  HEATHER   A        NICHOLLS  C01
PAOLOR9  JIMBO     J        BAEK      D11
```

Spiffy Computer Company wants to implement restrictions for employee's access from their remote applications.

11.3.2 Scenario implementation

To implement Spiffy Computer Company's requirements, the following procedures are planned:

- Creating identity mapping
- Creating a DB2 trusted context
- Creating a row permission
- Changing the application

Creating identity mapping

To create identity mapping in RACF, we need to define what kind of information will be sent from the remote application. Spiffy Computer Company uses the employee number as identity information and DB2 row permission applies to RACF user IDs, so we need set up mapping between empno (non RACF information) and the RACF user ID.

Example 11-18 shows the identity mapping in RACF.

Example 11-18 Identity mapping in the RACF

```
RACMAP ID(WEBUSER) MAP USERDIDFILTER(NAME('*')) REGISTRY(NAME('*'))
   WITHLABEL('Default user map from the remote')

RACMAP ID(PAOLOR1) MAP USERDIDFILTER(NAME('ERNIE')) REGISTRY(NAME('*'))
   WITHLABEL('ERNIE id map from the remote')

RACMAP ID(PAOLOR3) MAP USERDIDFILTER(NAME('EILEEN')) REGISTRY(NAME('*'))
   WITHLABEL('EILEEN id map from the remote')

RACMAP ID(PAOLOR5) MAP USERDIDFILTER(NAME('SEAN')) REGISTRY(NAME('*'))
   WITHLABEL('SEAN id map from the remote')

RACMAP ID(PAOLOR6) MAP USERDIDFILTER(NAME('EVA')) REGISTRY(NAME('*'))
  WITHLABEL('EVA id map from the remote')

RACMAP ID(PAOLOR7) MAP USERDIDFILTER(NAME('HEATHER')) REGISTRY(NAME('*'))
  WITHLABEL('HEATHER id map from the remote')
```

```
RACMAP ID(PAOLOR8) MAP USERDIDFILTER(NAME('CHISTINE')) REGISTRY(NAME('*'))
  WITHLABEL('CHISTINE id map from the remote')

RACMAP ID(PAOLOR9) MAP USERDIDFILTER(NAME('JIMBO')) REGISTRY(NAME('*'))
  WITHLABEL('JIMBO id map from the remote')
```

If you have many employees and not all of them need to have a RACF user ID, then it is a best practice to have a default identity mapping such as WEBUSER with a restricted default privilege. But a user should be authenticated before accessing a web server such as LDAP. If you have an employee who is not defined in the RACF, you maybe receive the `ICH408I DISTRIBUTED IDENTITY IS NOT DEFINED` error if you do not have the default WEBUSER identity mapping. The requirements of the customer environment determine how you handle a user without identity mapping.

Important: When you define identity propagation mapping, the identity name, registry name, and label are case sensitive.

Creating a DB2 trusted context

Now the DB2 administrator needs to create a DB2 trusted context between the web application server and DB2 for z/OS. Example 11-19 lists the DDL for the trusted context.

Example 11-19 Trusted context definition

```
CREATE TRUSTED CONTEXT CTXNAME1
       BASED UPON CONNECTION USING SYSTEM AUTHID DB2R7
       ATTRIBUTES (ADDRESS '9.30.28.119')
       ENABLE                                 ;
```

Creating a row permission

Now the DB2 administrator needs to create database access control by using the CREATE PERMISSION command, which was introduced in DB2 10.

Example 11-20 controls the employee table data. It forces employees accessing the table to only read their own information.

Example 11-20 Employee table access control

```
CREATE PERMISSION SECHEAD.ROW_EMP_RULES ON SPF.EMP FOR ROWS
WHERE (EMPNO = SESSION_USER)
ENFORCED FOR ALL ACCESS
ENABLE ;
```

More information about data access control can be found in Chapter 5, “Data access control” on page 87.

Changing the application

The application developer needs to change the Spiffy Computer Company’s web application. The developer creates a trusted connection using the CTXNAME1 trusted context through the DB2R7 user ID. The real user’s first name needs to be used because first name is the identity mapping rule.

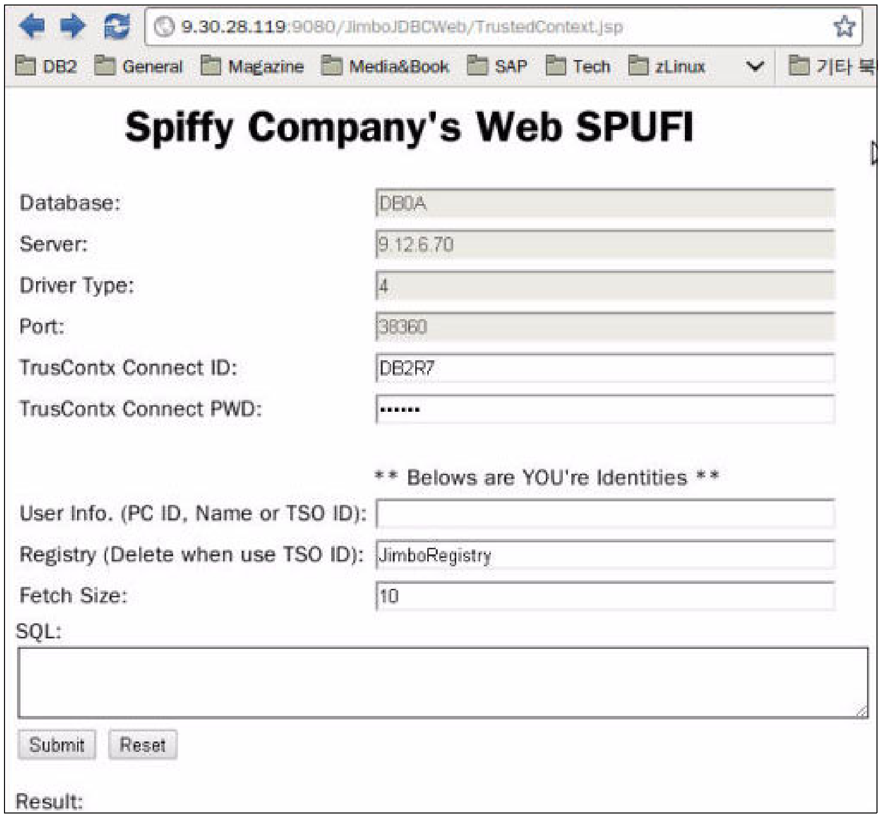
Example 11-21 shows the key application change for this implementation.

Example 11-21 Application change for trusted connection and identity propagation

```
   -------------------- Define DB2 connection Information --------------------

      DB2ConnectionPoolDataSource dataSource = new DB2ConnectionPoolDataSource();
      dataSource.setDatabaseName(databaseName);
      dataSource.setServerName(serverName);
      dataSource.setDriverType(Integer.valueOf(driverType));
      dataSource.setPortNumber(Integer.valueOf(portNumber));


   ------------------------ Connect trusted connection ------------------------

      Object[] objects = dataSource.getDB2TrustedPooledConnection(user, password,
properties);
      PooledConnection pooledCon = (PooledConnection)objects[0];


   ---- Change trusted connection user to RACF identity mapping information ----

       properties = new Properties();
      byte[] cookie = (byte[])objects[1];

      byte[] userSecTkn = null;

      String originalUser = null;

      String newpassword = null;

      con = ((com.ibm.db2.jcc.DB2PooledConnection)pooledCon).

getDB2Connection(cookie,newuser,newpassword,userRegistry,userSecTkn,originalUser,
properties);
```

The Spiffy Computer Company’s employee table data access control with DB2 10 supported RACF identity propagation is now ready.

Figure 11-16 shows the login window for the Spiffy Computer Company's web application.



9.30.28.119:9080/JimboJDBCWeb/TrustedContext.jsp

Spiffy Company's Web SPUFI

Database:	DB0A
Server:	9.12.6.70
Driver Type:	4
Port:	38360
TrusContx Connect ID:	DB2R7
TrusContx Connect PWD:	••••••
	** Belows are YOU're Identities **
User Info. (PC ID, Name or TSO ID):	
Registry (Delete when use TSO ID):	JimboRegistry
Fetch Size:	10

SQL:

Result:

Figure 11-16 Spiffy Computer Company's web application

Database access properties are set for the DB0A database and TrusContx Connect ID DB2R7.

When an employee enters the First Name in the User Info. field and enters the appropriate identity propagation registry information in the Registry field, the SQL is allowed to run against the table.

The trusted connection is first checked with the DB2R7 ID, which has the server IP address, and then the trusted connection ID change takes place according to the user identity information.

Figure 11-17 shows the Spiffy Computer Company's web SPUFI application flow diagram.

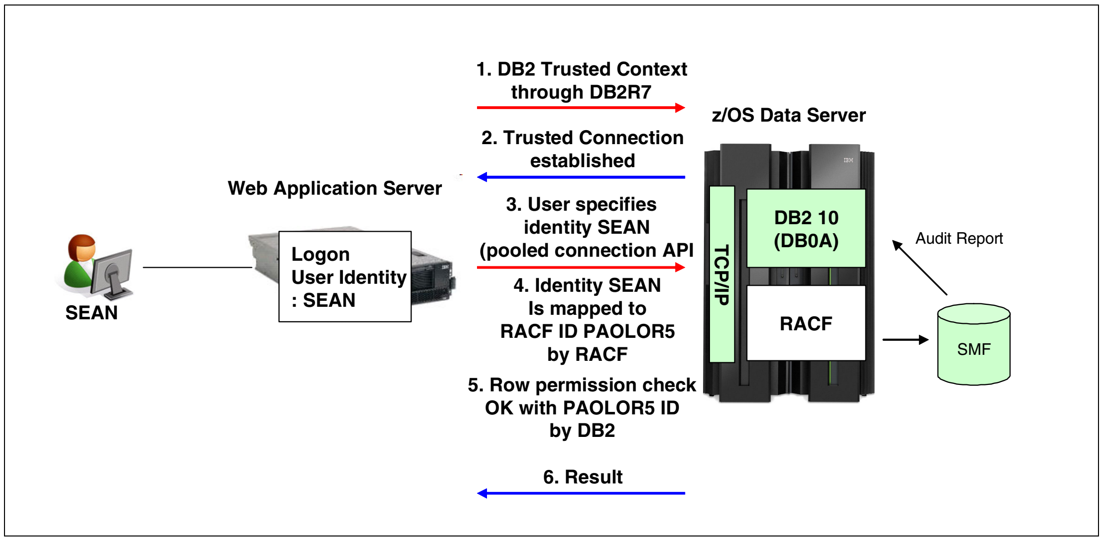
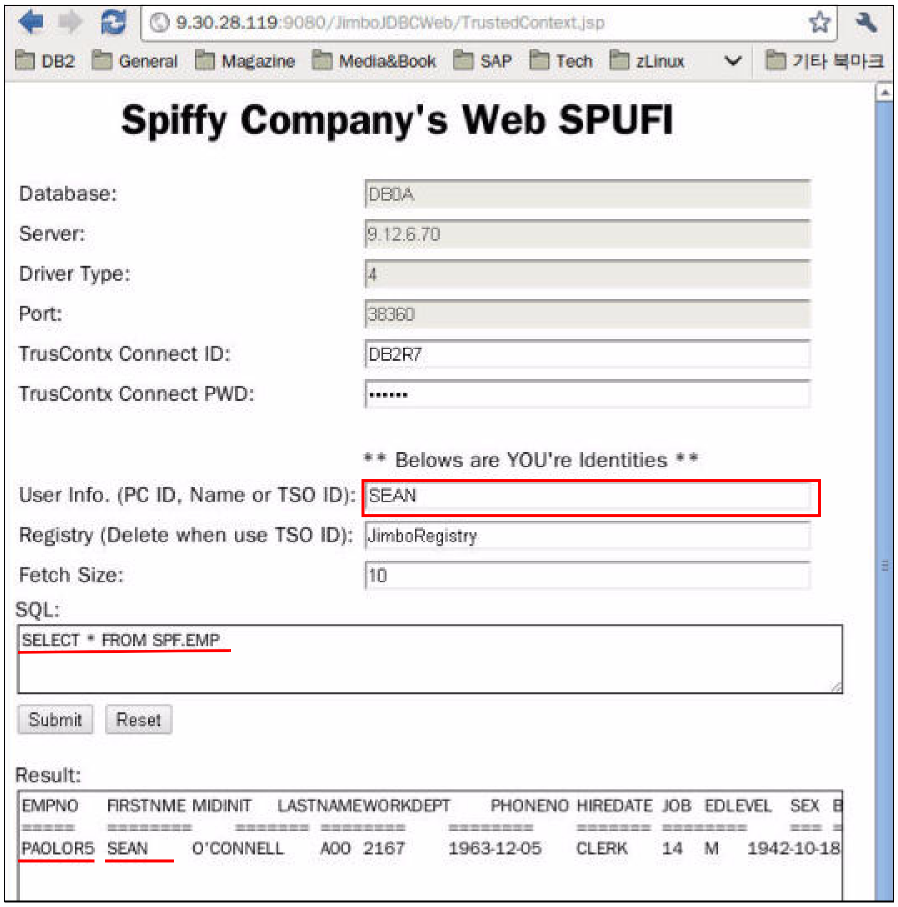

1. DB2 Trusted Context through DB2R7
2. Trusted Connection established
3. User specifies identity SEAN (pooled connection API
4. Identity SEAN Is mapped to RACF ID PAOLOR5 by RACF
5. Row permission check OK with PAOLOR5 ID by DB2
6. Result
z/OS Data Server
Web Application Server
SEAN
Logon User Identity : SEAN
TCP/IP
DB2 10 (DB0A)
RACF
Audit Report
SMF

Figure 11-17 Spiffy Computer Company application flow

We follow the steps in the diagram and run the application. The result is shown in Figure 11-18.



Spiffy Company's Web SPUFI

Database:	DB0A
Server:	9.12.6.70
Driver Type:	4
Port:	38360
TrusContx Connect ID:	DB2R7
TrusContx Connect PWD:	••••••

** Belows are YOU're Identities **

User Info. (PC ID, Name or TSO ID):	SEAN
Registry (Delete when use TSO ID):	JimboRegistry
Fetch Size:	10

SQL:

```
SELECT * FROM SPF.EMP
```

Result:

```
EMPNO   FIRSTNME MIDINIT   LASTNAMEWORKDEPT   PHONENO HIREDATE JOB  EDLEVEL  SEX  B
=====   ========  =======  ========  ========  =======  ========    ===  =
PAOLOR5 SEAN      O'CONNELL   A00  2167   1963-12-05  CLERK   14   M   1942-10-18
```

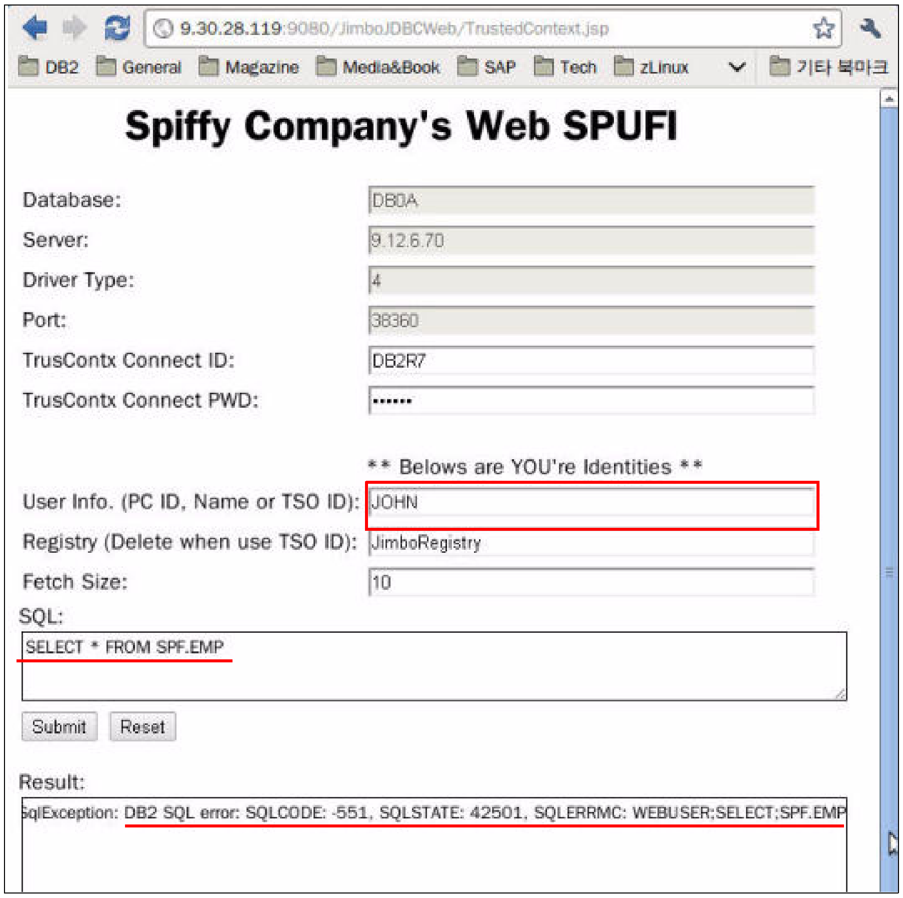
Figure 11-18 Remote client application's identity propagation and row permission result

We can see the result of employee table as selected by SEAN. SEAN is selecting from the employee table using his identity information, but identity propagation mapping lets DB2 know who SEAN is and DB2 then applies the row permission data access control.

In this case, identity propagation mapping for all the registries is using an asterisk as a wildcard, so any registry information is OK. In real life, use a specific registry name by policy.

Let us see what happens if we use another user identity who is not defined in the RACF identity propagation mapping. John is a new employee and there is no information in the employee table or RACF identity propagation mapping about him.

We can see the result of running an application using John's identity in Figure 11-19.



Spiffy Company's Web SPUFI

Database: DB0A
Server: 9.12.6.70
Driver Type: 4
Port: 38360
TrusContx Connect ID: DB2R7
TrusContx Connect PWD: ••••••

** Belows are YOU're Identities **
User Info. (PC ID, Name or TSO ID): JOHN
Registry (Delete when use TSO ID): JimboRegistry
Fetch Size: 10
SQL:

```
SELECT * FROM SPF.EMP
```

Result:

```
SqlException: DB2 SQL error: SQLCODE: -551, SQLSTATE: 42501, SQLERRMC: WEBUSER;SELECT;SPF.EMP
```

Figure 11-19 Result of non-identity propagation mapping user's access

The John identity is using WEBUSER, which is the default RACF user ID for remote access, and it has no privileges to select the SPF.EMP table.

Finally, we want to check the audit report. Auditing the DB2 remote client application is important, and we need to know who requested data from DB2, which privilege was used, what kind of unprivileged access was attempted and how many times it was tried, and so on.

Figure 11-20 shows the DB2 audit report after trying to make an authorization change to SEAN. We can see that the previous authorization ID was DB2R7, who is the DB2 trusted context authorization ID, and that the original authorization ID is SEAN. DB2 knows SEAN’s system ID is PAOLOR5 because of identity mapping, so DB2 allows SEAN to change to the PAOLOR5 primary authorization ID and work with PAOLOR5’s privileges.

```
1   LOCATION: DB0A                             OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)              PAGE: 1-2
       GROUP: N/P                                   AUDIT REPORT - DETAIL                       REQUESTED FROM: 05/23/11 19:36:00.00
      MEMBER: N/P                                                                                           TO: 05/23/11 23:59:00.00
   SUBSYSTEM: DB0A                                ORDER: PRIMAUTH-PLANNAME                         ACTUAL FROM: 05/23/11 19:36:30.00
 DB2 VERSION: V10                                       SCOPE: MEMBER                                       TO: 05/23/11 19:36:45.17
OPRIMAUTH CORRNAME CONNTYPE
 ORIGAUTH CORRNMBR INSTANCE
 PLANNAME CONNECT               TIMESTAMP   TYPE                                   DETAIL
 -------- -------- ------------ ----------- -------- --------------------------------------------------------------------------------
 PAOLOR5  db2jcc_a DRDA         19:36:30.09 AUTHCHG  TYPE:               REUSE TRUSTED CONTEXT           STATUS:           SUCCESS
 SEAN     ppli     C7D06DF492D2                      OBJECT OWNER:       AUTHID                          SQLCODE:                0
 DISTSERV SERVER                                     SECURITY LABEL:
 REQLOC  :::9.30.28.119                              CONTEXT NAME:       CTXNAME1
 ENDUSER :DB2R7                                      CONTEXT ROLE:
 WSNAME  :brenda                                     USER ROLE:
 TRANSACT:db2jcc_application                         PREV. SYSAUTHID:    DB2R7
                                                     REUSE AUTHID:       PAOLOR5
                                                     SERVAUTH NAME:
                                                     JOB NAME:
                                                     ENCRYPTION:
                                                     TCP/IP USED:
```

Figure 11-20 DB2 audit report for identity propagation from the remote application

We can audit and monitor using the previous authorization id and the original and primary authorization against the remote access workloads. For reference, brenda is a WebSphere Application Server’s node name and CTXNAME1 is trusted context name.

We can see one more audit report where JOHN, who is the default identity mapping user, was trying to access the employee table shown in Figure 11-21.

```
1   LOCATION: DB0A                             OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)              PAGE: 1-2
       GROUP: N/P                                   AUDIT REPORT - DETAIL                       REQUESTED FROM: 05/23/11 19:36:00.00
      MEMBER: N/P                                                                                           TO: 05/23/11 23:59:00.00
   SUBSYSTEM: DB0A                                ORDER: PRIMAUTH-PLANNAME                         ACTUAL FROM: 05/23/11 19:36:30.00
 DB2 VERSION: V10                                       SCOPE: MEMBER                                       TO: 05/23/11 19:36:45.17
OPRIMAUTH CORRNAME CONNTYPE
 ORIGAUTH CORRNMBR INSTANCE
 PLANNAME CONNECT               TIMESTAMP   TYPE                                   DETAIL
 -------- -------- ------------ ----------- -------- --------------------------------------------------------------------------------
 WEBUSER  db2jcc_a DRDA         19:36:45.16 AUTHCHG  TYPE:               REUSE TRUSTED CONTEXT           STATUS:           SUCCESS
 JOHN     ppli     C7D06E02F38B                      OBJECT OWNER:       AUTHID                          SQLCODE:                0
 DISTSERV SERVER                                     SECURITY LABEL:
 REQLOC  :::9.30.28.119                              CONTEXT NAME:       CTXNAME1
 ENDUSER :DB2R7                                      CONTEXT ROLE:
 WSNAME  :brenda                                     USER ROLE:
 TRANSACT:db2jcc_application                         PREV. SYSAUTHID:    DB2R7
                                                     REUSE AUTHID:       WEBUSER
                                                     SERVAUTH NAME:
                                                     JOB NAME:
                                                     ENCRYPTION:
                                                     TCP/IP USED:

 WEBUSER  db2jcc_a DRDA         19:36:45.17 AUTHFAIL AUTHID CHECKED: WEBUSER            PRIVILEGE: SELECT
 JOHN     ppli     C7D06E02F38B                      OBJECT TYPE   : TAB/VIEW           REASON:       0 RC: -    1
 DISTSERV SERVER                                     SOURCE OBJECT : EMP                SOURCE OWNER:   SPF
 REQLOC  :::9.30.28.119                              TARGET OBJECT : N/A                TARGET OWNER:   N/A
                                                     MLS    RID    : N/P                SECLABEL:       N/P
                                                     TEXT: SELECT * FROM SPF.EMP
```

Figure 11-21 DB2 audit report about authorization failure from a remote application

In this case, JOHN was trying to SELECT from the employee table but his ID was not an identity propagation mapping user, so he is running the SQL statement under the WEBUSER privilege.

11.3.3 Lessons learned

Today, many transactions that run on a DB2 subsystem originate outside of z/OS from the internet. They are initiated by users who authenticate their identities on web-based or distributed application servers, such as with Spiffy Computer Company.

The ability to monitor and identify remote users behind an application user ID was already provided by DB2 9 with trusted contexts and the “allow use for” option. With distributed identity filters, similar extended functionality and auditing is provided by both DB2 10 and RACF.

A distributed identity filter is a RACF mapping association between a RACF user ID and one or more distributed user identities. You can use the RACF RACMAP command to associate a distributed user identity with a RACF user ID.

DB2 10 supports the RACF distributed identity filter. Using this function, we can identify remote applications and we can have a better monitoring and auditing environment.

You need to look at the DB2 trusted context definition and the RACF mapping to know who is allowed to use the trusted connection.

11.4 Considerations about SQL injection

SQL injection is yet another common vulnerability that is the result of lax input validation. Unlike cross-site scripting vulnerabilities that are ultimately directed at your site’s visitors, SQL injection is an attack on the site itself.

The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or the user input is not strongly typed and thereby unexpectedly executed. It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another.

SQL injection attacks are also known as SQL insertion attacks.

We can avoid most SQL injection using several techniques, such as implementing secure coding best practices and limiting web application coding privileges, reducing debugging information, and testing web applications regularly. We can also use a product like pureQuery, IBM DataPower®, and so on. For more information about pureQuery, refer to 14.3, “SQL injection and IBM Optim pureQuery Runtime” on page 340.

11.4.1 Background information

Spiffy Computer Company needs to protect itself from SQL injection attacks by enhancing the remote dynamic SQL statement application’s security. So they review several product solutions and methodologies.

11.4.2 Implementation scenario

You need to understand the limitations of security within an application. System security can use security and integrity mechanisms that are not available to application programs. The level of assurance that can be provided in system security can be much higher. If the applications are run on the client or have fewer protection layers and firewalls than the database, make sure to address those limitations.

You should consider implementing the following functions:

- Input validation.
- Escaping characters.
- Encoding (input and output).
- Using prepared statements (for example, parameterized SQL queries).
- Accept numbers for a numeric comparison only.
- Do not allow special characters if they do not apply.
- Other options
  - Whitelists: A list of known allowed characters
  - Blacklists: A list of known bad characters that are never accepted

SQL injection attacks might occur when dynamic SQL statements are constructed from user input and the input is inadequately checked. You can use several techniques to prevent or reduce SQL injection attacks:

- Avoid dynamic SQL, whenever possible.
- Use SQLJ rather than JDBC for Java.
- Use system security techniques:
  - VIEWs
  - Trusted connections
  - Column masking
  - Row permissions

We have mentioned methodologies that protect against SQL injection, but we also need to discuss methodologies for detecting SQL injection. Because we cannot prevent SQL injection in every case, we also need to detect malicious error logs.

RACF message `ICH408I`, shown in Example 11-22, is the message that occurs when a failed user attempt against the RACF defined resources or policies occurs. The reason why we need to check this message is that we can discover failed attempts by identity mapping users that are not defined to log in to the remote DB2 application.

Example 11-22 RACF messages about logging in failure

```
ICH408I USER(PAOLOR5 ) GROUP(SYS1    ) NAME(P.                 ) 140
  LOGON/JOB INITIATION - INVALID PASSWORD ENTERED AT TERMINAL TCP67028

ICH408I USER(JIMBO   ) GROUP(        ) NAME(???                ) 535
  LOGON/JOB INITIATION - USER AT TERMINAL          NOT RACF-DEFINED
IRR012I  VERIFICATION FAILED. USER PROFILE NOT FOUND.

ICH408I USER(STC     ) GROUP(SYS1    ) NAME(STARTED TASK       ) 533
  DISTRIBUTED IDENTITY IS NOT DEFINED:
  X<4A>X<49>X<4D>X<42>X<4F> X<D1>X<89>X<94>X<82>X<96>X<D9>X<85>X<87>X<
89>X<A2>X<A3>X<99>X<A8>
IRR012I  VERIFICATION FAILED. USER PROFILE NOT FOUND.
```

Audit trace class 1 gives us the ability to detect the denied access attempts due to inadequate authorization. This trace is the most important trace when attempting to discover malicious DB2 users. A sample is shown in Figure 11-22.

```
OPRIMAUTH CORRNAME CONNTYPE
 ORIGAUTH CORRNMBR INSTANCE
 PLANNAME CONNECT                TIMESTAMP   TYPE                                    DETAIL
 -------- -------- ------------ ----------- -------- --------------------------------------------------------------------------------
 WEBUSER  db2jcc_a DRDA         19:55:06.43 AUTHFAIL AUTHID CHECKED: WEBUSER            PRIVILEGE: SELECT
 JIMBO    ppli     C7D2F5D802CD                      OBJECT TYPE    : TAB/VIEW          REASON:       0 RC: -    1
 DISTSERV SERVER                                     SOURCE OBJECT : SYSDATABASE        SOURCE OWNER:   SYSIBM
 REQLOC  :::9.30.28.119                              TARGET OBJECT : N/A                TARGET OWNER:   N/A
                                                     MLS    RID    : N/P                SECLABEL:       N/P
                                                     TEXT: SELECT * FROM SYSIBM.SYSDATABASE
 ENDUSER :DB2R7
 WSNAME  :brenda
 TRANSACT:db2jcc_application
```

Figure 11-22   Audit report for inadequate authorization

We need to check the SQLCODE in the application, as someone is attempting to maliciously discover object names, column names, and other object structure using SQL error codes and so on.

Product solutions

There is a filtering technique to prevent SQL injection where you filter input data before running an SQL statement, For example, WebSphere DataPower appliances can protect your servers from XML threats and SQL injection. SQL injection is the practice of submitting fields meant to contain data with values that contain SQL statements (for more information, refer to 14.3, "SQL injection and IBM Optim pureQuery Runtime" on page 340). Such values could cause the back-end application code to issue a database query different than the originally intended. In DataPower, SQL Injection Protection works by inspecting input fields for known patterns used by SQL injection attacks and rejecting the message if such a pattern is found.

Adding SQL Injection Protection requires extra testing to ensure that legitimate values are not rejected by mistake. If such rejections are encountered during testing, the SQL injection filter should be customized for the application.

All services that use payload fields to create SQL queries should be protected by a SQL injection filter. An example of a SQL injection filter is shown circled in red in Figure 11-23.

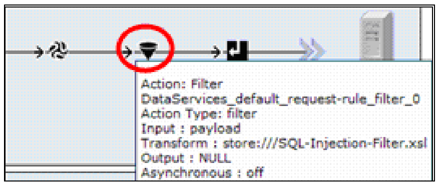

Action: Filter
DataServices_default_request-rule_filter_0
Action Type: filter
Input : payload
Transform : store:///SQL-Injection-Filter.xsl
Output : NULL
Asynchronous : off

Figure 11-23   DataPower SQL inject filtering

For additional protection against returning unauthorized responses from a service, business-level validation of the response should be performed. Business-level validation is different from schema validation in that it checks for occurrences of XML that may be valid from a schema perspective but unauthorized from a business perspective. An example would be a query for information about a single customer account that somehow returns information about multiple customer accounts. Such a response could be the result of a SQL injection attack that was not detected by the SQL injection filter.

Another technique is changing dynamic SQL to static SQL to prevent an SQL statement from changing. For example, pureQuery can greatly reduce the risk of SQL injection attacks by limiting SQL execution to only those SQL statements that appear in its SQL file. This task can be accomplished without changing applications and can be used with any Java framework.

As an example, you can use pureQuery to reduce vulnerability to SQL injection attacks by capturing the application SQL. The application SQL can be captured while programming using Data Studio Developer, be generated from a WebSphere Application Server JPA object, or be captured from existing Java applications while they are executing in "capture" mode on a test or development system. After the SQL is captured, it becomes easy to share and review the SQL with development colleagues or database administrators.

The captured SQL can be replaced in the capture file without going back to change the application. This action is particularly helpful for fixing performance issues such as those introduced by application frameworks or inexperienced developers. The Data Studio Developer edit windows validates that the new SQL is equivalent to the original to ensure the correct execution of the application.

Optionally, the SQL can be bound for static execution. Static execution has many advantages, including many relating to security. The static execution model grants access to a set of SQL statements instead of granting access to the table directly, thus limiting database access privileges, part of security best practices.

Finally, you can restrict SQL execution to only those SQL statements included in the captured SQL file. Any attempt to execute a statement not in the file will be rejected.

From the SQLCODE monitoring point of view, an SQL statement monitoring product, for example, IBM Query Monitor, can show every SQL error code from the applications and load them in a database, where we can analyze them easily.

Lessons learned

Several methodologies can prevent from SQL injection, but new SQL injection attack methods are being developed all the time. So it is impossible to prevent from all possible attacks.

No methodology can avoid all SQL injection, so we need to protect against SQL injection by using security log monitoring as well.

12

Database monitoring and the audit application

In this chapter, we describe several ways in which to use audit event information for report generation and activity tracking.

This chapter contains the following topics:

- Activity monitoring options on DB2 for z/OS
- Tivoli OMEGAMON for DB2 Performance Expert Version V5R1

12.1 Activity monitoring options on DB2 for z/OS

As shown in Chapter 8, “Audit policies” on page 151, DB2 10 for z/OS provides audit policies, a more secure mechanism for controlling the use of audit trace, and helps satisfy concerns regarding the separation of audit event collection from system and database administrators. This is a key point for demonstrating an uncontaminated audit event collection and reporting mechanisms as required to comply to internal and external requirements.

However, after the data is collected, some form of reporting capability needs to be in place to turn the raw audit event details into a more consumable form, something that is easily understood by non-technical security and audit compliance members of your organization.

Some customers create in-house processes that use the audit event data and generate reports for auditing, and in many cases, due to the nature of these processes, require direct and ongoing support and intervention of the database technicians. This involvement still represents a separation of role concern, and any reporting process needs to be maintained and well understood by the security and audit department without outside involvement of privileged users such as database and system administrators.

As we described in Chapter 8, “Audit policies” on page 151, after the audit event data is collected, you can use Tivoli OMEGAMON XE for DB2 Performance Expert Version V5R1 (OMEGAMON PE) to generate various audit reports. There are other third-party performance and SMF reporting products that are commonly used to provide similar functionality. For our purposes, we choose to use OMEGAMON PE.

12.2 Tivoli OMEGAMON for DB2 Performance Expert Version V5R1

OMEGAMON PE is a performance analysis, monitoring, and tuning tool for DB2 for z/OS environments. This product is part of the integrated and complete cross System z monitoring solution of the Tivoli OMEGAMON XE family that monitors all DB2 subsystems on z/OS and other resources, such as IMS, MVS, or CICS.

OMEGAMON PE simplifies and supports system and application performance monitoring, reporting, trend analysis, chargeback usage, and buffer pool analysis. If problems are encountered, you are notified and advised how to continue.

You can also use OMEGAMON PE to collect and evaluate historical data for different tasks. Historical data is useful for tuning, for problem analysis, for trend analysis, and for capacity planning. Included in OMEGAMON PE is the Performance Database. This is a set of tables that can be filled with DB2 performance information collected through SMF or a batch job using the FPEZCRD program. You can then retrieve, aggregate, or filter the information using SQL. To load the collected DB2 information into the performance database, functions of the reporter component (batch engine) have to be used.

The OMEGAMON PE Reporter generates predefined reports to help you collect and analyze historical performance data. It also enables you to prepare performance data before you load it into the Performance Warehouse or into the Performance Database.

The Reporter comes with the following predefined reports:

- Accounting
- Statistics
- System Parameters

- Utility
- Locking and Audit
- I/O Activity
- Record Trace
- SQL Activity

You can also use a report language to filter, sort, and group the data in the reports according to your preferences. For example, you can include or exclude specific data, sort or summarize by various options, and enable or disable reporting about specific performance data. You can use the User-Tailored Reporting (UTR) function to even further customize the reports.

You can use the Reporter to generate reports in the following ways:

- By submitting a batch job.
- By using the Interactive Report Facility (IRF). You can define the report commands and input data through ISPF panels. You can also submit the reports to run in the background.
- By using the Performance Warehouse Client. You can specify the report commands and input data through a graphical user interface. You can also submit the jobs immediately or schedule them to run later. This method only applies to ACCOUNTING and STATISTICS reports.

When you want to load data into the Performance Warehouse or the Performance Database, you can use the command language in the Reporter to indicate which data should be prepared and to indicate how the data should be summarized. You can process the data in the following ways:

- By submitting a batch job.
- By using the Performance Warehouse Client to either submit the job immediately or to schedule the job to run later.

You can use one of the following methods to collect the data that the Reporter uses:

- A batch job. In this case, you use the FPEZCRD program to collect performance data from a DB2 subsystem.
- The Performance Warehouse Client. In this case, you can use a graphical user interface to configure the job and to either submit it or to schedule it to run later.
- The Collect Report Data function in the ISPF Monitoring Dialogs. In this case, you can use ISPF panels to configure the job and to submit it.
- SMF or GTF data sets.
- Sequential data sets generated by Near-Term-History Data Collector configured to store trace data to VSAM data sets and sequential data sets (VSAMSEQ).

12.2.1 Executing the OMEGAMON PE batch reporter

The following scenario describes a representative use of the OMEGAMON PE batch reporter to generate audit reports. To limit the number of events collected, and to ensure they only represent our sample workload, we elect to use GTF.

To start the GTF trace, complete the following steps:

1. Start the GTF trace, either by using the MVS console command or a JCL procedure. On our system, we used the JCL shown in Example 12-1 to start the GTF trace.

   Example 12-1 GTF sample procedure

   ```
   //GTFDB2  PROC MEMBER=GTFDB2U
   //IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
   //  TIME=1440,REGION=2880K
   //IEFRDER DD   DSNAME=DB2R5.GTFDB2.&SYSNAME..D&YYMMDD..T&HHMMSS,
   //     DISP=(,CATLG),SPACE=(CYL,100,RLSE),UNIT=SYSDA
   //SYSLIB  DD   DSNAME=SYS1.PARMLIB(&MEMBER),DISP=SHR
   ```

   The GTF trace start parameters need to specify the GTF trace record type used by DB2 for tracing. This is coded in the JCL shown in Example 12-1 in the SYSLIB DD statement. These parameters are shown in Example 12-2.

   Example 12-2 GTF trace parameters

   ```
   TRACE=USRP
    USR=(FB9)
   ```

2. After the trace is started, either using the command or running the GTF sample JCL procedure, GTF indicates the allocation of the data set used to contain the trace records. This is described by the DD statement IEFRDER, shown in Example 12-1.

   For more information about GTF, refer to *z/OS V1R11 MVS Diagnosis Tools and Service Aids*, GA22-7589.

3. The DB2 trace needs to be activated. For our sample scenario, we use AUDITPOLICY to start a trace, specifying the audit category of EXECUTE for two tables. "Creating audit policy samples" on page 158 shows the sample SQL to create and insert policies into SYSIBM.SYSAUDITPOLICIES.

4. After the audit policy rows have been inserted, the audit policy trace is started. In our scenario, we specify a destination of GTF. In our example, we want to audit two tables, so we define two separate policies and start them with a single command, as shown in Example 12-3.

   Example 12-3 Starting multiple policies with a single start trace command

   ```
   -STA TRACE(AUDIT) DEST(GTF) AUDTPLCY(AUDERNIEEMP,AUDERNIEEMP2)
   ```

5. We then execute a small representative workload against our audited tables. One set of SQL statements was executed using a static bound package associated with a COBOL application. Example 12-4 shows the code segment used.

   Example 12-4 COBOL SQL sample

   ```
   PERFORM 3 TIMES
      SET EMP       TO TRUE
      PERFORM 3 TIMES
         EXEC SQL
            SELECT COUNT(*) INTO :V-EMP FROM DSN81010.EMP
         END-EXEC
         PERFORM U00-Handle-Result
      END-PERFORM
      SET DEPT TO TRUE
      PERFORM 3 TIMES
         EXEC SQL
   ```

```
        SELECT COUNT(*) INTO :V-EMP FROM DSN81010.DEPT
      END-EXEC
      PERFORM U00-Handle-Result
   END-PERFORM
   EXEC SQL COMMIT END-EXEC
   DISPLAY 'Commit performed'
END-PERFORM.
.
```

The other set of statements were executed dynamically using the DSNTEP2 sample program. These SQL statements are shown in Example 12-5.

Example 12-5 Sample dynamic SQL statements

```
INSERT INTO DSN81010.EMP
 VALUES (   '300000' , 'ERNIE' , ' ' , 'MANCILL' , 'A00' , '1234' ,
 '1998-08-29' , 'ITJOB' , 42 , 'M' , '1958-09-13' , 99.99 , 99.99 , 578)
 ;
COMMIT;
INSERT INTO DSN81010.EMP
 VALUES (   '300001' , 'ERNIE' , ' ' , 'REDFISH' , 'A00' , '1234' ,
 '1998-08-29' , 'ITJOB' , 42 , 'M' , '1958-09-13' , 99.99 , 99.99 , 578)
 ;
COMMIT;
 DELETE FROM DSN81010.EMP WHERE EMPNO >= '300000';
 COMMIT  ;
```

6. We stop the GTF trace collection, as described in *z/OS V1R11 MVS Diagnosis Tools and Service Aids*, GA22-7589. Looking at the output from the stop GTF console command, the generated data set, as defined in the JCL shown in Example 12-1 on page 278, is displayed.
7. After the audit trace event data set was made available, we submit the job in Example 12-6 to generate the OMEGAMON PE batch reporter. The sample JCL and report parameters are shown.

Example 12-6 OMEGAMON PE sample Audit Report

```
//DB2PE    EXEC PGM=DB2PM
//STEPLIB  DD  DISP=SHR,DSN=OMEGAXE4.TKANMOD
//STEPLIB  DD  DISP=SHR,DSN=OMEGASYS.SC63.XEDB2.XEDB2RTE.RKANMOD
//INPUTDD  DD  DISP=SHR,DSN=DB2R1.GTFDB2.SC63.D110518.T235430
//DPMLOG   DD  SYSOUT=*
//JOBSUMDD DD  SYSOUT=*
//STRPTDD  DD  SYSOUT=*
//ACRPTDD  DD  SYSOUT=*
//AUDITDD  DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
AUDIT
            REPORT
            LEVEL(DETAIL)
                     DDNAME(AUDITDD)
EXEC
/*
```

For more information about the use of the OMEGAMON PE batch reporter, refer to *IBM Tivoli OMEGAMON XE for DB2 Performance Monitor for z/OS Version 5.1 Report Reference,* SH12-6921.

After the report is executed, the audit report in Example 12-7 shows the static statement flow.

Example 12-7 Sample audit report

```
PRIMAUTH CORRNAME CONNTYPE
ORIGAUTH CORRNMBR INSTANCE
PLANNAME CONNECT                TIMESTAMP   TYPE                                DETAIL
-------- -------- ------------ ----------- -------- ------------------------------------------------------------
DB2R1    DB2R1R06 TSO          23:55:22.47 DML      TYPE    : 1ST READ
DB2R1    'BLANK'  C7CA5E7E45ED                      DATABASE: DSN8D10A            TABLE OBID:     18
AUDSQL   BATCH                                      PAGESET : DSN8S10E            LOG RBA   : X'000000000000'

DB2R1    DB2R1R06 TSO          23:55:22.50 DML      TYPE    : 1ST READ
DB2R1    'BLANK'  C7CA5E7E45ED                      DATABASE: DSN8D10A            TABLE OBID:     11
AUDSQL   BATCH                                      PAGESET : DSN8S10D            LOG RBA   : X'000000000000'

DB2R1    DB2R1R06 TSO          23:55:22.50 DML      TYPE    : 1ST READ
DB2R1    'BLANK'  C7CA5E7E45ED                      DATABASE: DSN8D10A            TABLE OBID:     18
AUDSQL   BATCH                                      PAGESET : DSN8S10E            LOG RBA   : X'000000000000'
```

Notice that the database and pageset identifiers have been translated, which is due to the fact that we started IFCID 105, which enables the OMEGAMON PE object translation. However, these identifiers are database and table space names, but the table name is not translated.

12.2.2 Loading event data into DB2 tables SQL based reporting

As shown in Example 12-7, SQL events can be shown in the OMEGAMON PE audit report. However, when combining multiple types of information about an audit event, it becomes difficult to achieve this goal within the confines of the OMEGAMON PE audit report set. Consider the following scenarios where relying on the OMEGAMON PE reporter could be viewed as cumbersome:

- Reporting requirements where combining data from multiple audit event types is required.
- Reporting requirements where combining data from different sources, for example, joining event data from audit policy collection with information from the DB2 catalog for OBID/PSID translation.
- Support for long term retention and management of audit event data. With many of the current regulations, audit data retention rules require the storage of many months of event data for reporting.
- Support for the ability to use a number of different processes (tools) to process the event data and to generate reports.
- Placing the audit data into a centralized repository to provide a central point of access control, with the ability to restrict who can access and modify the contents of the audit event repository.

For all of the above listed reasons, many customers choose to use the OMEGAMON PE batch reporter FILE utility to format the audit event trace records into a format that can be input into the DB2 10 LOAD utility. After it is loaded into relational format, then the data can be processed in meaningful ways.

12.2.3 OMEGAMON PE Performance Database

You can use the sample statements in the sample library to create and load a Performance Database. If you use an SMP/E sharing runtime environment, the sample library is TKO2SAMP. For all other types of runtime environments, the sample library is RKO2SAMP.

The Performance Database is a DB2 database that can hold aggregated and raw DB2 activity information spanning a long period of time. The type of data stored in the Performance Database depends on the reporter command (FILE or SAVE) used to create input data sets for the DB2 LOAD utility. This long-term history can help you with performance tuning activities, with trend analysis, and with capacity planning. Many customers already have existing instances of the OMEGAMON PE performance database, and in order to provide for segregation of audit data from other types of performance related information, they might consider creating a separate database in which to define the audit related tables of the OMEGAMON PE Performance Database.

Table 12-1 describes the RKO2SAMP members that contain DDL to define the various audit tables contained in the OMEGAMON PE Performance Database.

Table 12-1 RKO2SAMP DDL members to create audit tables

Table description	SAMPLIB member
Bind non repeating sections	DGOCXBND
Bind repeating sections	DGOCXBRD
Authorization changes	DGOCXCHG
DDL events	DGOCXDDL
Authorization control	DGOCXCNT
Authorization failures	DGOCXFAI
SQL statements	DGOCXSQL
Utility events	DGOCXUTI
DML events	DGOCXDML

When creating these objects, you need to carefully consider the size of the table spaces, depending on your particular auditing objectives, long term storage requirements, and level of audited activity on the target DB2 subsystem. These objects could require significant levels of storage.

After these objects are created, the next step in the population of these tables require the collection of audit events, either using the START TRACE AUDIT approach, or by using DB2 10 audit policies. Typically, customers direct the output of these traces to the IBM System Measurement Facility (SMF) data set.

SMF is a component of z/OS, providing a standardized method for writing out records of activity to a data set. SMF provides full *instrumentation* of all baseline activities running on a z/OS mainframe operating system, including I/O, network activity, software usage, error conditions, processor utilization, and so on.

SMF records are generated by hundreds of collecting routines spread all over the z/OS components and other software products. In general, SMF records describe the use of system resources by transactions. All the created SMF records are sent to I/O buffers in the SMF address space. When those buffers become full, a write I/O operation is executed, moving the records to z/OS VSAM data sets (SYS1.MANx). You have to allocate and catalog such data sets (at least two) on a high performance 3390 device because, if the rate of record creation is higher than the write I/O rate, you lose SMF records.

IBM provides a utility program (IFASMFDP) that copies (dumps) the contents of a full SYS1.MANx data set to a tape or to a DASD data set. After such copy completes, the SMF data set is flagged as empty and ready for a new load of SMF records.

SMF forms the basis for many monitoring and automation utilities. Each SMF record has a numbered type; in the case of DB2 audit trace events, these are associated with SMF record type 102. For more information about SMF, refer to *ABC's of z/OS System Programming Volume 2,* SG24-6982.

The dumped SMF data sets now contain the SMF events collected by the trace facility described above. This data can be designed as input into the OMEGAMON PE reporter. Example 12-8 shows sample JCL and the OMEGAMON PE reporter FILE command parameters.

Example 12-8 Sample PE reporter JCL and FILE parameter

```
//STEPLIB  DD  DISP=SHR,DSN=OMEGASYS.SC63.XEDB2.XEDB2RTE.RKANMOD
//INPUTDD  DD  DISP=SHR,DSN=DB2R1.GTFDB2.SC63.D110520.T161852
//DPMLOG   DD  SYSOUT=*
//JOBSUMDD DD  SYSOUT=*
//STRPTDD  DD  SYSOUT=*
//ACRPTDD  DD  SYSOUT=*
//AUDITDD  DD  SYSOUT=*
//AUFILDD1 DD  DISP=(NEW,CATLG,DELETE),
//             DSN=DB2R1.AUFILDD1.SYSREC,
//             DCB=(RECFM=VB,LRECL=9072,BLKSIZE=9076),
//             SPACE=(TRK,(500,500),RLSE),UNIT=SYSDA
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
AUDIT
            FILE TYPE(ALL)
                     DDNAME(AUFILDD1)
EXEC
/*
```

In Example 12-8, the input SMF data set is identified by the DD statement for INPUTDD, and the LOAD consumable output from the FILE process is identified by the DD statement AUFILDD1.

After it is processed, the FILE command output can then be used to load the different audit tables in the OMEGAMON PE performance database. Table 12-2 shows the sample LOAD utility control statements provided in RKO2SAMP for each of the performance database audit tables.

Table 12-2 RKO2SAMP LOAD utility control statements

Audit table name	Load utility statement
DB2PMFAUDT_DML	DGOXLDML
DB2PMFAUDT_BNDNR	DGOXLBNR
DB2PMFAUDT_BNDR	DGOXLBND
DB2PMFAUDT_AUTHCHG	DGOXLCHG
DB2PMFAUDT_AUTHCTR	DGOXLCTR
DB2PMFAUDT_DDL	DGOXLDDL
DB2PMFAUDT_AUTHFAI	DGOXLFAI
DB2PMFAUDT_SQLTEXT	DGOXLSQL
DB2PMFAUDT_UTILITY	DGOXLUTI

For more information about the use of the OMEGAMON PE batch reporter, and the FILE subcommand, refer to *IBM Tivoli OMEGAMON XE for DB2 Performance Monitor for z/OS Version 5.1 Report Reference,* SH12-6921.

12.2.4 Authorization failure reporting and loading events into DB2

This scenario describes an environment where all authorization failures are reported for auditing purposes. We inserted the following row into SYSIBM.SYSAUDITPOLICIES to turn on tracing, as shown in Example 12-9.

Example 12-9 Audit policy row for authorization failures

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME, CHECKING)
VALUES('POLICY_CLS_1_AUTHFAIL','A');
```

After the audit policy was activated, the following scenario was attempted:

- An insert into SYSIBM.SYSAUDITPOLICIES is performed by ID DB2R1, who is SYSADM, but does not have privileges to insert into policy table.
- SET CURRENT is issued by DB2R1 to attempt to connect to the SECHEAD secondary ID, which can update SYSIBM.SYSAUDITPOLICIES. SECHEAD is not in the RACF list of groups connected ID for DB2R1.
- There is an attempt to create a ROLE with authorization ID DB2R2 who is not SECADM.
- There is an attempt to create a ROLE with PAOLOR1 who is not SECADM.
- A SELECT from SYSIBM.SYSTABAUTH is performed by user PAOLOR7, who does not have SELECT privileges against the DB2 catalog objects.
- A SELECT from SPF.EMP table is performed by user XMLR1, who does not have SELECT privileges against this table.

We first execute the OMEGAMON PE batch reporter to generate audit events with the FILE command. Example 12-10 shows the command sequence used.

Example 12-10 DB2PM batch reporter FILE command

```
FPEC2001I  COMMAND INPUT FROM DDNAME SYSIN
           AUDIT
                      FILE TYPE(ALL)
                              DDNAME(AUFILDD1)
           EXEC
FPEC1999I  SYSTEM INITIALIZATION COMPLETE. RETURN CODE 0
FPEC0999I  EXECUTION COMPLETE. RETURN CODE 0
```

In examining the output, the IFCID distribution report shows the different events that were collected during our trace interval. Example 12-11 shows the distribution report.

Example 12-11 OMEGAMON PE batch report IFCID distribution

```
DB2 VERSION: V10                                                                                               TO: 05/20/11 16:29:00.00
             INPUT        INPUT       PROCESSED     PROCESSED                          INPUT        INPUT       PROCESSED     PROCESSED
   IFCID     COUNT     PCT OF TOTAL     COUNT     PCT OF TOTAL          IFCID      COUNT     PCT OF TOTAL     COUNT     PCT OF TOTAL
  -------  ----------  ------------  ----------  ------------         -------  ----------  ------------  ----------  ------------
      4             1      2.70%             1      2.77%               140             8     21.62%             8     22.22%
    105            27     72.97%            27     75.00%               362             1      2.70%             0      0.00%
     TOTAL INPUT TRACE RECORDS  =                37
     TOTAL PROCESSED TRACE RECORDS =             36
```

In reviewing this report, we see there were a total of eight authorization failures captured through IFCID 140 event data. In addition, we see a number of IFCID 105 records, which are due to the additional IFCID 105 trace being active, which is needed to allow for DBID/OBID translation in the event data to occur.

Next, we load these events into the DB2PM audit tables; the process for this task has been previously outlined in 12.2.3, "OMEGAMON PE Performance Database" on page 281.

An auditor might be interested in having a list of all attempts that resulted in security violations. The SQL query that can provide this information is shown in Example 12-12.

Example 12-12 SQL to generate a list of security violations

```
SELECT PRIMAUTH, CHAR(DATE(TIMESTAMP),USA), TIME(TIMESTAMP),
  PLAN_NAME, AUTHID_CHECKED, OBJECT_TYPE, PRIV_ATTEMPTED, SOURCE_OWNER,
 SOURCE_OBJECT, TARGET_OWNER, TARGET_OBJECT
 FROM DB2R1.DB2PMFAUDT_AUTHFAI
 WHERE NOT PLAN_NAME = ' '
 ORDER BY 1, 3;
```

When the SQL statement from Example 12-12 is executed, the report in Example 12-13 is generated.

Example 12-13 Report example for all authorization failures

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--
PRIMAUTH                        PLAN_NAME  AUTHID_CHECKED  OBJECT_TYPE  PRIV_ATTEMPTED   SOURCE_OWNER  SOURCE_OBJECT
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--
DB2R1     05/20/2011  16.22.58  DSNESPCS   DB2R1           TAB/VIEW     INSERT           SYSIBM        SYSAUDITPOLICIES
DB2R1     05/20/2011  16.23.35  DSNESPCS   DB2R1           USERAUTH     SYSADM
DB2R2     05/20/2011  16.23.29  DSNESPCS   DB2R2           ROLE                                        TEST_ROLE
PAOLOR1   05/20/2011  16.23.55  DSNESPCS   PAOLOR1         ROLE                                        TEST_ROLE
PAOLOR7   05/20/2011  16.25.55  ADB        PAOLOR7         USERAUTH     EXPLAIN          SYSIBM        SYSTABAUTH
PAOLOR7   05/20/2011  16.25.55  ADB        PAOLOR7         TAB/VIEW     SELECT           SYSIBM        SYSTABAUTH
XMLR1     05/20/2011  16.28.41  ADB        XMLR1           USERAUTH     EXPLAIN          SPF           EMP
XMLR1     05/20/2011  16.28.41  ADB        XMLR1           TAB/VIEW     SELECT           SPF           TARPON
```

As our scenario was previously outlined, the following authorization failure events were collected.

- DB2R1 attempted to INSERT a row into SYSIBM.SYSAUDITPOLICIES.
- DB2R1 attempted to use a SYSADM authority.
- DB2R2 attempted to create a ROLE named TEST_ROLE.
- PAOLOR1 was used in an attempt to create the ROLE named TEST_ROLE.
- PAOLOR7 attempted to use DB2 Administration Tool to EXPLAIN an SQL statement accessing SYSIBM.SYSTABAUTH.
- PAOLOR7 attempted to then SELECT from SYSIBM.SYSTABAUTH.
- XMLR1 attempted to use DB2 Administration Tool to EXPLAIN an SQL statement accessing SPF.EMP.
- XMLR1 then attempted to SELECT from SPF.EMP.

A word of explanation about the presence of the EXPLAIN statements in the authorization failure query is in order. When DB2 Administration Tool is used to browse the content of a DB2 table, which was our scenario, the statement is first explained to provide a cost estimate of the query to the Administration Tool. Although our expectation was to only see the SELECT statement, the IFCID 140 record was cut for both access attempts, SELECT and EXPLAIN.

In looking at this report, the auditor might next want to see the specific SQL statement associated with the authorization failure. The SQL statement text is contained in a separate audit table, DB2PMFAUDT_SQLTEXT, which can be correlated with the authorization failure event information contained in DB2PMFAUDT_AUTHFAI. Example 12-14 shows the SQL statement to generate this report snippet.

Example 12-14 SQL statement combining authorization failure to SQL statement text

```
SELECT X.PRIMAUTH, CHAR(DATE(X.TIMESTAMP),USA) AS DATE,
  TIME(X.TIMESTAMP) AS TIME,
  X.PLAN_NAME, X.SOURCE_OWNER, X.SOURCE_OBJECT,
  X.PRIV_ATTEMPTED,
 Y.STMT_TEXT
 FROM DB2R1.DB2PMFAUDT_AUTHFAI X, DB2R1.DB2PMFAUDT_SQLTEXT Y
 WHERE X.TIMESTAMP = Y.TIMESTAMP
 ORDER BY 1, 3;
```

Running this SQL query creates the report shown in Example 12-15.

Example 12-15 Authorization failure combined with SQL statement

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--------
PRIMAUTH  DATE        TIME      PLAN_NAME  SOURCE_OWNER  SOURCE_OBJECT       PRIV_ATTEMPTED   STMT_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--------
DB2R1     05/20/2011  16.22.58  DSNESPCS   SYSIBM        SYSAUDITPOLICIES    INSERT           INSERT INTO SYSIBM.SYSAUDITP
DB2R1     05/20/2011  16.23.35  DSNESPCS                                     SYSADM           SET CURRENT SQLID
DB2R2     05/20/2011  16.23.29  DSNESPCS                 TEST_ROLE                            CREATE ROLE TEST_ROLE
PAOLOR1   05/20/2011  16.23.55  DSNESPCS                 TEST_ROLE                            CREATE ROLE TEST_ROLE
PAOLOR7   05/20/2011  16.25.55  ADB        SYSIBM        SYSTABAUTH          SELECT           SELECT * FROM SYSIBM.SYSTABA
PAOLOR7   05/20/2011  16.25.55  ADB        SYSIBM        SYSTABAUTH          EXPLAIN          SELECT * FROM SYSIBM.SYSTABA
XMLR1     05/20/2011  16.28.41  ADB        SPF           EMP                 SELECT           SELECT * FROM "SPF"."EMP" FO
XMLR1     05/20/2011  16.28.41  ADB        SPF           EMP                 EXPLAIN          SELECT * FROM "SPF"."EMP" FO
DSNE610I NUMBER OF ROWS DISPLAYED IS 8
```

In a production environment, where the majority of access is through well protected applications (CICS, WebSphere, IMS), and most SQL is bound in static packages, authorization failures should be a rare event, and any occurrence should be examined, as access failures could represent evidence of external attacks being executed against the DB2 10 server. Another interesting report is a list of authorization failures for each authorization identifier. Running this report on a daily basis helps the auditor to establish a steady state understanding of the application behavior. Any significant deviation from the established norm, for example, a number of violations for an authorization ID used by an application server, could be evidence of someone attempting to use the application server credentials to access unauthorized resources. Example 12-16 shows the sample SQL for this query.

Example 12-16 SQL statement for number of failures by authorization ID

```
SELECT PRIMAUTH,
   AUTHID_CHECKED, COUNT(*)
   FROM DB2R1.DB2PMFAUDT_AUTHFAI
 GROUP BY PRIMAUTH, AUTHID_CHECKED
```

When executed, the SQL statement generates the report shown in Example 12-17.

Example 12-17 Count of authorization failures by authorization identifier

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+--------
PRIMAUTH  AUTHID_CHECKED
---------+---------+---------+---------+---------+---------+---------+---------+---------+--------
DB2R1     DB2R1                    2
DB2R2     DB2R2                    1
PAOLOR1   PAOLOR1                  1
PAOLOR7   PAOLOR7                  2
XMLR1     XMLR1                    2
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

When the auditors reviews the above query result, they might want to investigate the specifics of the activity associated with one authorization ID. In Example 12-17, we choose to look at XMLR1, which for the purposes of our discussion, might be associated with an application server.

As shown in Example 12-18, this SQL statement generates a list of failures for a single authorization ID.

Example 12-18 Authorization failures for a single authorization ID

```
SELECT PRIMAUTH, CHAR(DATE(TIMESTAMP),USA), TIME(TIMESTAMP),
  PLAN_NAME, AUTHID_CHECKED, OBJECT_TYPE, PRIV_ATTEMPTED, SOURCE_OWNER,
 SOURCE_OBJECT, TARGET_OWNER, TARGET_OBJECT
 FROM DB2R1.DB2PMFAUDT_AUTHFAI
 WHERE PRIMAUTH IN ('XMLR1')
 ORDER BY 1, 3;
```

The results from Example 12-18 on page 286 are shown in Example 12-19.

Example 12-19 Authorization failure for a single authorization ID

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+----
PRIMAUTH                        PLAN_NAME  AUTHID_CHECKED  OBJECT_TYPE  PRIV_ATTEMPTED   SOURCE_OWNER  SOURCE_OBJECT       T
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+----
XMLR1     05/20/2011  16.28.41  ADB        XMLR1           USERAUTH     EXPLAIN          SPF           EMP
XMLR1     05/20/2011  16.28.41  ADB        XMLR1           TAB/VIEW     SELECT           SPF           EMP
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
```

12.2.5 Monitoring and reporting on changes in the security environment

It is important to understand when changes occur to DB2 10 that affect the security environment. Introduced with the audit policy controls are the policy categories SECMAINT and VALIDATE. SECMAINT creates audit event information when the security administration functions either GRANT or REVOKE privileges. The VALIDATE category collects any authorization changes that occur, including the use of the SET CURRENT command. Example 12-20 shows the audit policy row to activate these audit policy categories. In this example, we also activate CHECKING, which collects authorization failures.

Example 12-20 Authorization change categories activated in audit policy

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
  (AUDITPOLICYNAME,
   CHECKING, SECMAINT, VALIDATE)
  VALUES('POLICY_CLS_127_EMP','A','A','A');
```

Our scenario is as follows:

1. Insert policy row.
2. Activate this policy using the START TRACE command.
3. Attempt to grant SYSADM to XMLR1 with PAOLOR1, which will fail, as we are running under SEPARATE SECURITY (Y) and PAOLOR1 lacks SECADM.
4. SET CURRENT SQLID to SECHEAD, which contains SECADM.
5. GRANT SYSADM privilege to XMLR1.
6. Revoke SELECT on SPF.EMP from XMLR1.

We start the trace with the command shown in Example 12-21.

Example 12-21 Start audit policy trace for VALIDATE, CHECKING, and SECMAINT

```
-STA TRACE(AUDIT) DEST(GTF) AUDTPLCY(POLICY_CLS_127_EMP)
```

After we execute the scenario, we process the audit events using OMEGAMON PE batch reporter, and then loaded these events into the audit tables.

One question that might be of interest to an auditor would be a list of GRANT or REVOKE statements being executed over a set period of time. To see this process, we combine the DB2PMFAUDT_AUTHCTR table, which contains the IFCID 141 event data with the DB2PMFAUDT_SQLTEXT table to show the SQL statement for the GRANT or REVOKE. Example 12-22 shows this query.

Example 12-22 Authorization changes with either GRANT or REVOKE

```
SELECT X.ORIGAUTH, CHAR(DATE(X.TIMESTAMP),USA) AS DATE,
  TIME(X.TIMESTAMP) AS TIME,
  X.AUTHCNTL_USERID, X.AUTHCNTL_SQLCODE AS SQLCODE,
  X.OBJECT_TYPE,
  Y.STMT_TEXT
 FROM DB2PMFAUDT_AUTHCTR X, DB2PMFAUDT_SQLTEXT Y
 WHERE X.TIMESTAMP = Y.TIMESTAMP
 ORDER BY 1, 3;
```

When we run this query, we see the result from the statement in Example 12-23.

Example 12-23 Authorization changes query result

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-------
ORIGAUTH  DATE       TIME      AUTHCNTL_USERID      SQLCODE  OBJECT_TYPE  STMT_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-------
PAOLOR1   05/21/2011  17.46.45  PAOLOR1                 -552  USERAUTH     GRANT SYSADM TO XMLR1
PAOLOR1   05/21/2011  17.47.06  SECHEAD                    0  USERAUTH     GRANT SYSADM TO XMLR1
PAOLOR1   05/21/2011  17.47.25  SECHEAD                    0  TAB/VIEW     REVOKE SELECT ON TABLE SPF.EMP FROM XMLR1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
```

Notice that the first event would also appear in the authorization failure reporting process, as it resulted in the SQLCODE -552 due to insufficient authorization for PAOLOR1.

The changing of authorization by the use of SET CURRENT SQLID is another area of audit reporting interest. This information is recorded in IFCID 55 and captured in the DB2PMFAUDT_AUTHCHG audit table. The SQL in Example 12-24 shows how to report on this event.

Example 12-24 SQL showing the use of SET CURRENT SQLID

```
SELECT ORIGAUTH, CHAR(DATE(TIMESTAMP),USA) AS DATE,
  TIME(TIMESTAMP) AS TIME,
  PRIMAUTH, CURRENT_AUTHID, '-->TRANSLATED INTO-->', NEW_AUTHID,
  STMT_STATUS
 FROM DB2PMFAUDT_AUTHCHG
 WHERE IFCID = '55'
 ORDER BY 1, 3;
```

After running the above query against the audit table, we see the report in Example 12-25.

Example 12-25 SQL query result showing list of SET CURRENT SQLID usages

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-------
ORIGAUTH  DATE       TIME      PRIMAUTH  CURRENT_AUTHID                        NEW_AUTHID  STMT_STATUS
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-------
PAOLOR1   05/21/2011  17.47.06  PAOLOR1   PAOLOR1        -->TRANSLATED INTO-->  SECHEAD     SUCCESS
PAOLOR1   05/21/2011  17.47.25  PAOLOR1   PAOLOR1        -->TRANSLATED INTO-->  SECHEAD     SUCCESS
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
```

One other piece of information that might be useful from an audit perspective is that when system privileges are granted to an authorization ID, this ID could be connected through RACF to other RACF IDs. When granting the RACF ID a higher level of DB2 authority, you might want to ensure that the ID is not connected through RACF list of group processing. In the scenario in Example 12-23, the ID XMLR1 might be a GROUP rather than an individual authorization ID.

In 9.3.8, “Using the RACF database unload utility IRRDBU00” on page 204, we discuss how to unload the content of the RACF database into DB2 tables. We can construct an SQL statement to validate that XMLR1 is a RACF ID or a GROUP. Example 12-26 shows the sample SQL statement.

Example 12-26 RACF database query for RACF ID definition

```
SELECT USBD_NAME,
USBD_CREATE_DATE,
USBD_OWNER_ID
 FROM DB2R1.USER_BD
 WHERE USBD_NAME = 'XMLR1'
 ORDER BY 1, 2;
```

The result from this query, if successful, shows that the DB2 authorization ID is associated with a specific RACF ID, as shown in Example 12-27.

Example 12-27 Query results for RACF ID SQL select

```
---------+---------+---------+---------+---------+---------+---------+---------+
USBD_NAME  USBD_CREATE_DATE  USBD_OWNER_ID
---------+---------+---------+---------+---------+---------+---------+---------+
XMLR1      2010-09-02        HAIMO
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
```

If this query results in row not found, then the authorization ID being granted is either a ROLE or a GROUP ID. To determine which one it is, and to see who is connected to this authorization ID, the query in Example 12-28 can be performed against the RACF database.

Example 12-28 RACF query to show if ID is Group

```
SELECT * FROM DB2R1.GROUP_BD
    WHERE GPBD_NAME LIKE 'SEC%';
```

After the query in Example 12-28 is executed, the result is shown in Example 12-29.

Example 12-29 Query results for RACF group SQL select

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+------
GPBD_NAME  GPBD_SUPGRP_ID  GPBD_CREATE_DATE  GPBD_OWNER_ID  GPBD_UACC  GPBD_NOTERMUACC  GPBD_INSTALL_DATA
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+------
SECADM     SYS1            2003-04-17        ANTOFF         NONE       N
SECHEAD    SYS1            2011-05-05        PAOLOR7        NONE       N
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
```

If the authorization ID is a group, then the SQL query in Example 12-30 shows the list of RACF IDs that are connected to this group.

Example 12-30 RACF query to show RACF ID connections to a GROUP

```
SELECT * FROM DB2R1.GROUP_MEMBERS
   WHERE GPMEM_NAME LIKE 'SEC%';
```

The query in Example 12-30 on page 289 produces the result shown in Example 12-31.

Example 12-31 RACF ID connected to a RACF GROUP

```
---------+---------+---------+---------+---------+---------+---
GPMEM_NAME  GPMEM_MEMBER_ID  GPMEM_AUTH
---------+---------+---------+---------+---------+---------+---
SECADM      DB2R1            USE
SECADM      DB2R2            USE
SECADM      DB2R3            USE
SECADM      DB2R4            USE
SECADM      DB2R6            USE
SECADM      DB2R7            USE
SECADM      DB2R8            USE
SECADM      DB2R9            USE
SECHEAD     PAOLOR1          USE
SECHEAD     PAOLOR2          USE
DSNE610I NUMBER OF ROWS DISPLAYED IS 10
```

When processing requests for DB2 authorities, in particular system privileges, the security administration function must ensure that there are no unintended proliferation of privileges through the presence of connected ROLE or GROUPs.

12.2.6 Monitoring the use of privileges in DB2

Prior to DB2 10, when the auditors wanted to monitor the activities of a privileged user, they needed to start the various trace classes with the AUTHID phrase. The subsequent audit events would be collected, based on the AUTHID filter, depending on the trace classes that were active. In some combinations, this situation might result in more data that was needed to satisfy the audit requirement. More importantly, as new authorization IDs are created, with system privileges, it is a requirement that the list of AUTHID identifiers in the START TRACE process would need to be updated to collect activities of these newly added authorization IDs with privileges. If the construction of the authorization scheme allows for the use of the wildcard in the filter (wildcard support was added with DB2 9), then there is less risk for missing an ID in the filter.

Introduced with DB2 10 is a new audit class specifically designed for monitoring the use of system privileges in DB2, which is the class 11 trace category. This category of tracing can also be controlled with the use of the audit policy category SYSADMIN and DBADMIN.

Example 12-32 shows the audit policy row to activate these audit policy categories. In this example, we also activate CHECKING, which collects authorization failures.

Example 12-32 SYSADMIN use categories activated in audit policy

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME,CHECKING,SYSADMIN,DBADMIN)
VALUES('POLICY_CLS_11_SYSADM2','A','*','*');
```

Our scenario is as follows:

1. Insert the policy row.
2. Activate this policy using the START TRACE command.
3. ALTER table SPF.EMP to AUDIT ALL.
4. ALTER table SPF.EMP to AUDIT NONE.
5. SELECT * from SPF.EMP.

We start the trace using the command shown in Example 12-33.

Example 12-33 Start policy trace for SYSADM

```
-STA TRACE(AUDIT) DEST(GTF) AUDTPLCY(POLICY_CLS_11_SYSADM2)
```

After the event data is collected, we then run the OMEGAMON PE batch reporter FILE utility to format the audit event data, and load this information into the audit tables. For this scenario we then run the SQL in Example 12-34 to generate the activity report for tracking use of administration privileges.

Example 12-34 SQL to collect events from SYSADMIN

```
SELECT X.PRIMAUTH, CHAR(DATE(X.TIMESTAMP),USA) AS DATE,
  TIME(X.TIMESTAMP) AS TIME,
  X.PLAN_NAME, X.ORIGAUTH, X.CORRNAME, X.CONNECT_TYPE,
 Y.STMT_TEXT
 FROM DB2R1.DB2PMFAUDT_AUTHCTR X, DB2R1.DB2PMFAUDT_SQLTEXT Y
 WHERE X.TIMESTAMP = Y.TIMESTAMP
 ORDER BY 1, 3;
```

The result from the query is shown in Example 12-35.

Example 12-35 Audit report for SYSADM activity

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+
--
PRIMAUTH  DATE       TIME      PLAN_NAME  ORIGAUTH  CORRNAME  CONNECT_TYPE  STMT_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+
--
DB2R1     05/23/2011  17.08.22  ADB       DB2R1     DB2R1     TSO           ALTER TABLE "SPF"."EMP" .. AUDIT ALL
DB2R1     05/23/2011  17.08.31  ADB       DB2R1     DB2R1     TSO           ALTER TABLE "SPF"."EMP" .. AUDIT NONE
DB2R1     05/23/2011  17.08.39  ADB       DB2R1     DB2R1     TSO           SELECT * FROM "SPF"."EMP" FOR FETCH ONLY
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
```

Finding the SQL statement text in static and dynamic statements

Finding the SQL statement text when executing an audit policy based trace is different depending on if the statement is from a STATIC or DYNAMIC request. Our first scenario is based on the use of a STATIC statement. We use the sample COBOL application that we described in “Sample scenario” on page 166.

First, we start the audit policies for DSN81010.EMP and DSN81010.DEPT, which were created earlier in “Creating audit policy samples” on page 158. Next, we execute our COBOL sample application, AUDSQL.

We then execute the OMEGAMON PE batch reporter using the command sequence in Example 12-36 to generate audit events with the FILE command.

Example 12-36 DB2PM batch reporter FILE command

```
FPEC2001I  COMMAND INPUT FROM DDNAME SYSIN
           AUDIT
                        FILE TYPE(ALL)
                                 DDNAME(AUFILDD1)
           EXEC
FPEC1999I  SYSTEM INITIALIZATION COMPLETE. RETURN CODE 0
FPEC0999I  EXECUTION COMPLETE. RETURN CODE 0
```

When we run the FILE command, examination of the IFCID distribution report shows the distribution report in Example 12-37.

Example 12-37 IFCID distribution report for static SQL scenario

```
LOCATION: DBOA                                    OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)
      GROUP: N/P                                    IFCID FREQUENCY DISTRIBUTION LOG
     MEMBER: N/P
  SUBSYSTEM: DBOA
DB2 VERSION: V10
              INPUT        INPUT      PROCESSED     PROCESSED                      INPUT
    IFCID     COUNT     PCT OF TOTAL    COUNT     PCT OF TOTAL          IFCID     COUNT
   -------  ----------  ------------  ----------  ------------         -------  ----------
     144            6     100.00%             6     100.00%
      TOTAL INPUT TRACE RECORDS  =               6
      TOTAL PROCESSED TRACE RECORDS =            6
```

After the audit event records have been created, we then load them into the OMEGAMON PE performance database. In looking at the resultant trace data, loaded into a table, the SQL in Example 12-38 can report on the activity associated with the bound application.

Example 12-38 Static SQL audit report SQL example

```
SELECT X.PRIMAUTH, CHAR(DATE(X.TIMESTAMP),USA) AS DATE,
TIME(X.TIMESTAMP) AS TIME,
X.PLAN_NAME, X.CONNECT, X.CORRNAME,
CASE X.IFCID
    WHEN 144 THEN 'SELECT'
    WHEN 143 THEN 'UPDATE'
             ELSE 'OTHER'
END AS STATEMENT,
Y.NAME AS TABLE_NAME
FROM DB2R1.DB2PMFAUDT_DML X, SYSIBM.SYSTABLES Y
WHERE (X.DATABASE_DBID = Y.DBID) AND (X.TABLE_OBID = Y.OBID)
ORDER BY 1, 3;
```

Running the above SQL statement generates the report shown in Example 12-39.

Example 12-39 Static SQL audit report output

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-
PRIMAUTH  DATE       TIME      PLAN_NAME  CONNECT   CORRNAME  STATEMENT  TABLE_NAME
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     DEPT
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     EMP
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     DEPT
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     EMP
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     DEPT
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     EMP
DSNE610I NUMBER OF ROWS DISPLAYED IS 6
```

When collecting the events using the audit policy category EXECUTE, we observe that while support for the DB2 10 statement identifier fields QW0143SI and QW0144SI have been added to the OMEGAMON PE trace record report, it does not appear in the audit report and is not supported in the audit FILE command at this time. As a result, it is not possible to correlate the static statement identifier with the bound statement text stored in SYSIBM.SYSPACKSTMT.

Important: DB2 10 statement identifier support is currently not enabled for OMEGAMON PE audit report set and the audit FILE command.

For more information about the statement identifier, refer to 8.1.5, “Unique statement identifier” on page 164.

12.2.7 Finding the dynamic statement

For dynamic statement execution, a slightly different approach is used to load and then report on the activity associated with dynamic SQL. The audit policy traces described in “Finding the SQL statement text in static and dynamic statements” on page 291 are still active. We then execute the SQL statements in Example 12-40 using DSNTEP2.

Example 12-40 Static SQL example

```
INSERT INTO DSN81010.EMP
 VALUES (   '300000' , 'ERNIE' , ' ' , 'MANCILL' , 'A00' , '1234' ,
 '1998-08-29' , 'ITJOB' , 42 , 'M' , '1958-09-13' , 99.99 , 99.99 , 578)
 ;
COMMIT;
INSERT INTO DSN81010.EMP
 VALUES (   '300001' , 'ERNIE' , ' ' , 'MANCILL' , 'A00' , '1234' ,
 '1998-08-29' , 'ITJOB' , 42 , 'M' , '1958-09-13' , 99.99 , 99.99 , 578)
 ;
COMMIT;
 DELETE FROM DSN81010.EMP WHERE EMPNO >= '300000';
 COMMIT  ;
```

We then collect the audit trace event records, and process them through OMEGAMON PE FILE command, then load these event details into the audit tables. When running the same query as described in Example 12-38 on page 292, we now see the rows in the DML audit table, as shown in Example 12-41.

Example 12-41 Audit report showing DML activity

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+------
PRIMAUTH  DATE       TIME      PLAN_NAME  CONNECT   CORRNAME  STATEMENT  TABLE_NAME
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+------
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     DEPT
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     EMP
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     DEPT
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     EMP
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     DEPT
DB2R1     05/23/2011  18.03.21  AUDSQL     BATCH     DB2R1R06  SELECT     EMP
DB2R1     05/23/2011  19.17.29  DSNTEP10   BATCH     DB2R1L    UPDATE     EMP
DB2R1     05/23/2011  19.17.29  DSNTEP10   BATCH     DB2R1L    UPDATE     DEPT
DB2R1     05/23/2011  19.17.29  DSNTEP10   BATCH     DB2R1L    UPDATE     EMP
DB2R1     05/23/2011  19.17.29  DSNTEP10   BATCH     DB2R1L    UPDATE     SNOOK
DSNE610I NUMBER OF ROWS DISPLAYED IS 10
```

For dynamic SQL, there is also information about the specific SQL statement, stored in the SQLSTMT table that we would also like to be able correlate with the DML table activity, but currently there is no easy way to perform this task because, as previously mentioned, support for the DB2 10 statement identifier fields QW0143SI and QW0144SI have not been added to the OMEGAMON PE AUDIT file command and the DML and SQLSTMT tables of the performance database. A time stamp is also not directly useful, as this is different between the IFCID 143/144 and the IFCID 145. If correlation is needed, one can approximate this correlation with the query in Example 12-42.

Example 12-42 Query to show statement text for dynamic SQL

```
SELECT PRIMAUTH, CHAR(DATE(TIMESTAMP),USA) AS DATE,
TIME(TIMESTAMP) AS TIME,
PLAN_NAME, IFCID, STMT_TEXT
FROM DB2R1.DB2PMFAUDT_SQLTEXT
ORDER BY 1, 3;
```

You could then manually correlate these events, as shown in Example 12-43, by plan name, authorization identifier, and date time.

Example 12-43 Audit report for SQL statement text for dynamic statements

```
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--------
PRIMAUTH  DATE        TIME      PLAN_NAME   IFCID  STMT_TEXT
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+--------
DB2R1     05/23/2011  19.17.29  DSNTEP10      145  INSERT INTO DSN81010.EMP VALUES ( '300000' ,
DB2R1     05/23/2011  19.17.29  DSNTEP10      145  INSERT INTO DSN81010.EMP VALUES ( '300001' ,
DB2R1     05/23/2011  19.17.29  DSNTEP10      145  DELETE FROM DSN81010.EMP WHERE EMPNO >= '3000
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
```

Even in a robust environment, for the dynamic statement, this manual correlation might prove sufficient for general audit reporting purposes.

12.2.8 Considerations for reporting using the OMEGAMON PE audit tables

Using the OMEGAMON PE audit tables to form a basis for audit reporting provides a flexible mechanism for storing and management of audit event data. When audit event data is stored in relational form, it provides the auditor the flexibility of using many different types of SQL based reporting tools. As we demonstrated, when combined with loading the RACF database content, along with RACF related SMF event information, audit reporting correlation between these different event sources is possible.

Depending on audit reporting requirements, the class of audit event data being collected, and data retention requirements, the amount of data collected can be significant. When designing for an audit reporting infrastructure based on DB2 tables, there are operational considerations that must be considered.

Most customers perform a nightly SMF process, typically where the individual SMF offload data sets are merged into a single file, and then processed. As part of this nightly process, using the IFASMFDP utility, the SMF type 102 records from the audit trace should be separated into a separate file. This data set should be RACF protected to prevent any unauthorized access or to preclude any manipulation of the audit event data. In addition, execution of the SMF offload process should be managed and controlled by the installation job scheduling and tracking system. Also, the IFASMFDP control data set that contains the offload parameters needs to be controlled by RACF, to prevent any changes to the SMF dump processing that might eliminate audit events.

As mentioned in “Compressing your audit records” on page 170, the use of z/OS compression services when creating the audit event records can result in some significant savings in the amount of storage consumed by the DB2 type 102 SMF record. Note that these savings come at the expense of a small increase in CPU.

After the nightly offload process is completed, you then need to run the OMEGAMON PE performance reporter FILE utility to convert the SMF type 102 records into DB2 LOAD utility consumable data. You can issue up to six report commands in a single OMEGAMON XE DB2 report execution. You should use one execution to avoid having to make multiple passes of the data to convert the SMF 102 with the FILE command. You might also consider breaking up the FILE commands to separately process certain event types, and, depending on your audit requirements and IFCID distribution characteristics, this action could allow you to perform the FILE conversion process and the LOAD utility execution in parallel.

Example 12-44 shows an example of how the OMPEGAMON XE DB2 FILE command could be constructed.

Example 12-44 FILE command with separate DD for each AUDIT type

```
AUDIT
      FILE TYPE(AUTHCHG)
               DDNAME(AUFILDD1)
      FILE TYPE(AUTHCNTL)
               DDNAME(AUFILDD2)
      FILE TYPE(AUTHFAIL)
               DDNAME(AUFILDD3)
      FILE TYPE(BIND)
               DDNAME(AUFILDD4)
      FILE TYPE(DDL)
               DDNAME(AUFILDD5)
      FILE TYPE(DML)
               DDNAME(AUFILDD6)
      FILE TYPE(UTILITY)
               DDNAME(AUFILDD7)
  EXEC
```

After the audit event data has been loaded into DB2, you should consider treating this information as any other critical operational database. To ensure adequate query performance, you should consider generating frequent catalog statistics through the use of RUNSTATS to improve the performance of any ad hoc query processes. Additionally, you should plan for unexpected failures and have implemented a strategy for backup and recovery of this data. Included in this planning should be the capability for recovering this information during unplanned offsite disaster contingency and recovery situations.

Another consideration, particularly when writing SQL statements that combine columns from multiple tables, is join performance. Commonly executed SQL statements should be EXPLAINed for maximum performance, and for frequently access tables, additional indexes should be considered to improve performance. By default, the audit tables in the samples supplied with OMEGAMON XE DB2 do not include any indexes.

Due to the sensitive nature of these audit tables and event information, to better manage the separation of roles and responsibilities concerns, more clearly demonstrate adequate controls, and to prevent any intentional change to the audit data, some customers create a separate DB2 subsystem to host the audit tables and the audit reporting infrastructure. These subsystems are created with a minimum number of privileged authorization identifiers, which are tightly controlled by the security administration function. This segregation provides the additional benefit of minimizing the impact to shared DB2 resources for other applications in a shared subsystem environment.

Finally, in most customer environments, audit event data retention becomes a concern. Given the long term retention requirements called for in many regulatory requirements, audit data needs to be retained for a long as 7 years. For the customer whose audit requirements include keeping this much event data, the auditing application can quickly become one of the larger DB2 applications in their environment. For such a customer, part of the planning for a DB2 z/OS based audit storage and reporting solution needs to include some capability to partition and archive inactive audit event data, but to also plan for subsequent retrieval and report generation in the event of an unanticipated long term reporting requirement. Some customers find that the functionality provided by the *IBM Optim Data Growth Solution*, which creates the framework for archiving inactive data, a valuable tool in addressing this issue.

13

DB2 temporal support

Industry regulations, compliance and auditing requirements, and competitive pressures are prompting IT managers to maintain more data for longer periods of time and to provide better ways for business users to analyze past, current, and future events.

To help organizations achieve these goals, IBM has built greater awareness of time into DB2 10 for z/OS. New DB2 temporal data management technology enables firms to track and query historical, current, and future conditions in a straightforward and efficient manner. The result is a simpler way to implement auditing and compliance initiatives, to pinpoint (and correct) human errors, to ensure the integrity of data over time, and to assess changing business conditions, such as insurance coverage, product prices, medical prescriptions, and interest rates on credit cards.

DB2 10 provides the capability to implement table-level specifications to control the management of data based on time. A temporal table is a table that records the period of time when a row is valid. DB2 supports three types of temporal tables. The term bitemporal is used when both periods are included:

- System-period temporal table
- Application-period temporal table
- Bitemporal table

This significant functionality can be defined inside DB2, rather than coding the same functionality in multiple places within an application.

DB2 10 brings many new security options. Temporal tables can also provide new security functionality by enabling the capture of historical information about data. Use of this function reduces development time, thus allowing you to meet compliance laws faster as DB2 now manages the historic versions of the data.

With temporal table support, programmers can write queries that specify a search criteria based on the time that data existed. DB2 10 provides unique, industry-leading database advanced business analytical capabilities that are easy to implement without significant coding effort.

Using system period temporal tables can help you automatically migrate data from a current table to a history table without changing the application.

In this chapter, we provide an introduction to temporal support and show some examples of temporal tables. This tutorial is entirely based on the contents of the white paper “A Matter of Time: Temporal Data Management in DB2 for z/OS” by Saracco, et al, and can be downloaded from the following address:

ftp://aix.software.ibm.com/software/data/sw-library/db2/zos/A_Matter_of_Time_-_DB2_zOS_Temporal_Tables_-_White_Paper_v1.4.1.pdf

We then show how a temporal table can enrich the solution for an auditing scenario.

This chapter contains the following topics:

- Temporal tables
- Using temporal tables for auditing

13.1 Temporal tables

Managing different versions of application data has been a struggle for programmers and DBAs for a long time. New regulations require maintaining historical versions of data for years, data updates, and row deletion events need to be recorded in history tables. Using the application to handle versioning through triggers or intricate logic adds complexity to the code and the table design. Maintaining the rules within the application implies documenting and sharing those rules with staff that changes over time or even for every application release. Bottom line, applying the rules within the application is error prone and likely inconsistent across applications.

Database built-in support for managing multiple versions of data and tracking effective business dates can save database administrators and application developers considerable time and effort.

Here are some sample scenarios:

- An internal audit requires a financial institution to report on changes made to a client’s records during the past 5 years.
- A pending lawsuit prompts a hospital to reassess its knowledge of a patient’s medical condition just before a new treatment was ordered.
- A client challenges an insurance firm’s resolution of a claim involving a car accident. The insurance firm needs to determine the policy’s terms in effect when the accident occurred.
- An online travel agency wants to detect inconsistencies in itineraries. For example, if someone books a hotel in Rome for eight days and reserves a car in New York for three of those days, the agency would like to flag the situation for review.
- A retailer needs to ensure that no more than one discount is offered for a given product during any period of time.
- A client inquiry reveals a data entry error involving the three-month introductory interest rate on a credit card. The bank needs to retroactively correct the error (and compute a new balance, if necessary).

For each of these situations, time is critical. The new DB2 temporal data management support helps implement time-aware applications and queries with minimal effort.

13.1.1 How temporal tables work

A system-period temporal table involves tracking when changes are made to the state of a table, such as when an insurance policy is modified or a loan is created.

An application-period temporal table involves tracking the effective dates of certain business conditions, such as the terms of an insurance policy or the interest rate of a loan.

These are the two types of time periods that DB2 supports:

- System period: A pair of columns with system-maintained values that indicate the period of time when a row is valid. A system period is named SYSTEM_TIME and it is intended for use in a system-period temporal table. The SYSTEM_TIME begin and end columns must be defined as:
  - `TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN`
  - `TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END`
- Application period: A pair of columns with application-maintained values that indicate the period of time when a row is valid. An application period is named BUSINESS_TIME and it is intended for use in an application-period temporal table. The begin and end columns can be defined as either:
  - `TIMESTAMP(6) WITHOUT TIME ZONE NOT NULL`
  - `DATE NOT NULL`

A unique index can be defined for a BUSINESS_TIME period to enforce non-overlapping time periods for the instances of a particular object modelled in the temporal table.

Each of the two period types is geared towards supporting a specific temporal capability. The system period is intended for supporting a concept of data versioning and is used for providing the system-maintained history of the data.

The application period is intended for supporting a user-controlled way of establishing the notion of validly of the data. It allows user applications to specify and control the periods when certain data is considered valid to the user.

DB2 uses an inclusive, exclusive approach for modelling time periods. The begin column contains the value for when a row is valid from. The end column contains the value for when a row stops being valid. The period's start time is included in the period but its end time is not. So, if the end time (TIMESTAMP(6) WITHOUT TIME ZONE NOT NULL) of an insurance policy was recorded as midnight on 31 December 2007, the policy would have been active until midnight (but not at midnight). And more importantly, if an end DATE (DATE NOT NULL) was recorded as 31 December 2007, then the insurance would have been only valid until 30 December 2007, not on 31 December 2007.

A bitemporal table includes both a system-period and an application-period:

- With system-period temporal tables, DB2 captures historical versions of rows. You now can have access to current or historical versions of rows.
- WIth application-period temporal tables, applications can manage future, current, and past versions of rows.

System-period data versioning can be implemented without affecting the application.

DB2 10 provides the capability to state table-level specifications to control the management of application data based upon time. Application programmers can specify search criteria based on the time the data existed or was valid. This capability simplifies DB2 application development requiring data versioning.

System-period temporal table

A system-period temporal table is implemented by altering an existing table to contain a SYSTEM _TIME period, or creating a new table with a SYSTEM_TIME period as follows:

1. Altering an existing table or creating a table with a SYSTEM_TIME period
2. Creating a history table
3. Defining the versioning relationship.

A SYSTEM_TIME period consists of a pair of columns with system-maintained values that indicate the period of time when a row is valid. System-period data versioning specifies that old rows are archived into another table. The table that contains the current active rows of a table is called the system-period temporal table. The table that contains the archived rows is called the history table. You can delete the rows from the history table when those rows are no longer needed.

A system-period temporal table must also have a transaction-start-ID column. A transaction-start-ID column specifies that a time stamp value is assigned when the row is inserted or any column in the row is updated. It allows you to associate the same time (the time of the first DML statement) to all the changes performed within a transaction.

In 13.1.2, “Creating a system-period temporal table” on page 301, we show an example of creating a table with system time.

Application-period temporal table

Business time is sometimes referred to as *valid time* or *application time*. An application period is a period in which you maintain the beginning and ending values for a row. The begin column contains the value for when a row is valid from. The end column contains the value for when a row stops being valid.

An application-period temporal table is implemented by altering an existing table to contain a BUSINESS_TIME period, or creating a new table with a BUSINESS_TIME period.

Bitemporal tables

A bitemporal table is a table that is both a system-period temporal table and an application-period temporal table. You can use a bitemporal table to keep application period information and system-based historical information. Therefore, you have a lot of flexibility in how you query data based on periods of time.

Performance expectations

For system-period temporal tables, SELECT of current data would perform similar to access to tables that are not defined for data versioning. DELETE and UPDATE of current data would perform slower than for a table not defined with data versioning. SELECT of historical data requires retrieving data from two tables.

For application-period temporal tables, a 1 - 3% impact for maintaining business periods without overlaps can be expected while DB2-provided business time support for row splitting outperforms the use of a user-defined stored procedure by 57% - 68% in DB2 CPU time.

In general, DB2 system-period and application-period temporal tables support will greatly reduce cost and labor required for application development. For a sample scenario studied, it was observed that both SYSTEM_TIME and BUSINESS_TIME support outperforms equivalent functions implemented with user-defined triggers or stored procedures.

For details about performance, see section 7.1, “Temporal support” of *DB2 10 for z/OS Performance Topics*, SG24-7942.

13.1.2 Creating a system-period temporal table

DB2 support for system time enables you to automatically track and manage multiple versions of your data. Specifically, by defining a table with a SYSTEM_TIME period, you are instructing DB2 to automatically capture changes made to the state of your table and to save *old* rows in a history table, which is a separate table with the same structure as your current table. Temporal queries referencing your current table cause DB2 to transparently access the history table when needed, as you will see shortly. This feature enables you to work with historical data easily, avoiding the need for complex WHERE clauses with various time stamp and join conditions.

Defining a system-period temporal table involves three simple steps:

1. Create the base table for current data.

   Include three TIMESTAMP(12) columns, that is, two for the start and end times of the SYSTEM_TIME period and one for the transaction-start-ID column:

   ```
   P_FROM TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN
   P_TO TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END
   TRANS_ID TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID
   PERIOD SYSTEM TIME (P_FROM, P_TO)
   ```

   DB2 uses the transaction start time to track when the transaction first executed a statement that changes the table's data and setting the row-begin column. You define all three TIMESTAMP columns as GENERATED ALWAYS so that DB2 automatically generates these values on INSERT, UPDATE, and DELETE. This definition relieves you from specifying values for these columns when writing to the database and also ensures that the time stamps are accurate. Optionally, you may define any of these three columns as IMPLICITLY HIDDEN so that they will not show up in SELECT * statements.

2. Create the history table.

   Define the structure of this table to correspond to the table containing the current data. You can achieve this easily by using a CREATE TABLE . . . LIKE statement.

3. Use an ALTER TABLE statement with the ADD VERSIONING and USE HISTORY TABLE clauses.

Let us step through an example to see how to instruct DB2 to automatically maintain multiple versions of your data using system time:

1. Create a table with a SYSTEM_TIME period. In our example, we specify that the trans_start column should be hidden. We show the DDL for a SYSTEM_TIME temporal table in Example 13-1.

   Example 13-1 DDL for a SYSTEM_TIME temporal table

   ```
   CREATE TABLE policy (
   id INT primary key not null,
   vin VARCHAR(10),
   annual_mileage INT,
   rental_car CHAR(1),
   coverage_amt INT,
   sys_start TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL,
   sys_end TIMESTAMP(12) GENERATED ALWAYS AS ROW END NOT NULL,
   trans_start TIMESTAMP(12) GENERATED ALWAYS
   AS TRANSACTION START ID IMPLICITLY HIDDEN,
   PERIOD SYSTEM_TIME (sys_start, sys_end)
   );
   ```

2. Create an associated history table:

   ```
   CREATE TABLE policy_history like policy;
   ```

3. Enable versioning:

   ```
   ALTER TABLE policy ADD VERSIONING USE HISTORY TABLE policy_history;
   ```

Table 13-1 and Table 13-2 show the two empty tables created as a result. Throughout this section, we depict the contents of the history table in grey to help you distinguish it from the current table.

Table 13-1 POLICY table (contains current data)

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end

Table 13-2 POLICY_HISTORY table (contains historical data)

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end

You can also use the ALTER TABLE statement to modify existing tables to track system time. To do so, you would need to add the appropriate TIMESTAMP(12) columns and add the PERIOD SYSTEM_TIME clause.

Inserting data into a system-period temporal table

Inserting data into a system-period temporal table is not any different from inserting data into an ordinary table. For example, imagine that on 18 May 2011, you need to enter two new car insurance policy records into your POLICY table. The statements in Example 13-2 accomplish this task.

Example 13-2 Insert two new policy rows

```
INSERT INTO policy(id, vin, annual_mileage, rental_car, coverage_amt)
VALUES(1111, 'A1111', 10000, 'Y', 500000);
INSERT INTO policy(id, vin, annual_mileage, rental_car, coverage_amt)
VALUES(1414, 'B7777', 14000, 'N', 750000);
```

When inserting each row into the current table, DB2 generates the appropriate TIMESTAMP(12) values for columns of the SYSTEM_TIME period and transaction-start-ID columns. Note that none of these were referenced in the INSERT statements; DB2 automatically records the necessary information. Table 13-3 and Table 13-4 on page 303 shows the contents of the POLICY and POLICY_HISTORY tables as a result of this query.

To make our example easier to follow, Table 13-3 and Table 13-4 on page 303 show only the date portion of the required TIMESTAMP(12) columns. Dates appear in YYYY-MM-DD format.

Table 13-3 POLICY table (contains current data) after inserts on 18 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	10000	Y	500000	2011-05-18	9999-12-30

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1414	B7777	14000	N	750000	2011-05-18	9999-12-30

Table 13-4 POLICY_HISTORY table (historical data) after inserts on 18 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end

The sys_start values in the POLICY table reflect when the rows were inserted (on 18 May 2011, in our example). The sys_end values are set to a high time stamp value by DB2. Effectively, this indicates that these rows have not expired, that is, the rows contain current data.

Updating data in a system-period temporal table

When you update current data, DB2 automatically maintains an old version of the data in the appropriate history table. This action happens transparently without any programming or user effort.

Imagine that the statement in Example 14-3 executes on 19 May 2011 to change the coverage amount for Policy 1111 from 500000 to 750000.

Example 13-3 Update a policy

```
UPDATE policy
SET coverage_amt = 750000
WHERE id = 1111;
```

Let us explore how DB2 processes this statement. As shown in Table 13-5 on page 304 and Table 13-6 on page 304, DB2 updates the value of the row in the current table. In addition, it moves a copy of the old row to the history table. For both tables, DB2 correctly records the sys_start and sys_end values for these rows. In particular, DB2 sets the sys_end column value for this row in the history table to the time of the transaction that issued the UPDATE statement. All this occurs transparently to the user. Although not shown in Table 13-5 on page 304 and Table 13-6 on page 304, DB2 also records the transaction-start-ID in both tables. The transaction-start-ID column is a generated column that is defined with the TRANSACTION START ID clause. The value is assigned whenever a row is inserted into a system-period temporal table or any column in the row is updated.

The transaction-start-ID time is the same as the system end time for the version of policy 1111 that is in the history table. This is because when the update was initiated, this version (500000) of the policy became history. At the same moment in time the new version (750000) became the current one. The transaction-start-ID time is the same for both the current row and the historical row, as both rows were affected by the same transaction, in this case a single update statement.

In effect, the processing of this UPDATE statement records that Policy 1111 had a coverage amount of 500000 set from 18 May 2011 to 19 May 2011; thereafter, the coverage amount was set to 750000.

Again, to make our example easier to follow, Table 13-5 and Table 13-6 show only the date portion of the required TIMESTAMP(12) columns.

Table 13-5 POLICY table current content after update on 19 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	10000	Y	750000	2011-05-19	9999-12-30
1414	B7777	14000	N	750000	2011-05-18	9999-12-30

Table 13-6 POLICY_HISTORY table current content after update on 19 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	10000	Y	500000	2011-05-18	2011-05-19

As you might expect, any subsequent updates to policies are handled in a similar manner. For example, assume that Policy 1111 is updated again on 19 May 2011, just a few minutes later, to change several details of the insurance policy, such as the annual mileage estimate, a rental car coverage, and overall coverage amount. The corresponding UPDATE statement is shown in Example 13-4.

Example 13-4 More policy updates on 19 May 2011

```
UPDATE policy
SET annual_mileage = 5000, rental_car='N', coverage_amt = 250000
WHERE id = 1111;
```

The update statement is successful. Note that number of rows affected is 1:

```
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

Executing this statement causes DB2 to automatically modify the POLICY and POLICY_HISTORY tables, as shown in Table 13-7 and Table 13-8.

Table 13-7 Current table content after UPDATE on 19 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	5000	N	250000	2011-05-19-13.30.46.194814065000	9999-12-30-00.00.00.000000000000
1414	B7777	14000	N	750000	2011-05-18-22.56.06.249020907000	9999-12-30-00.00.00.000000000000

Table 13-8 Historical table content after UPDATE on 19 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	10000	Y	500000	2011-05-18-22.56.06.249020907000	2011-05-19-13.23.19.707478331000

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	10000	Y	750000	2011-05-19-13.23.19.707478331000	2011-05-19-13.30.46.194814065000

We now show the full sys_start and sys_end time stamp values as the update happens on the same day as the first update.

At a high level, Table 13-7 on page 304 and Table 13-8 on page 304 are showing that:

- The original version of policy 1111 was valid from 2011-05-18-22.56.06.249020907000 to just before 2011-05-19 -13.23.19.707478331000.
- The next chronological version of policy 1111 was valid from 2011-05-19-13.23.19.707478331000 to just before 2011-05-19-13.30.46.194814065000.

The current version of policy 1111 is valid from 2011-05-19-13.30.46.194814065000 going forward.

Remember that the period's start time is included in the period but the end time is not.

Note: APAR PM31314 has modified the row-end maximum value to take into account the java.sql.Timestamp which is based on java.util.Date (where hours have values from 0 to 23) and the time zone calculation. This is reflected in our examples where the row-end maximum values is 9999-12-30-00.00.00.000000000000 instead of the original 9999-12-31-24.00.00.000000000000.

Deleting data from a system-period temporal table

When you delete current data, DB2 automatically removes the data from the current table and maintains an old version of the data in the appropriate history table. DB2 sets the end time of the (deleted) data in the history table to the transaction start time of the DELETE statement. This happens transparently without any programming or user effort. As you will see shortly, users can access this deleted data (that is, the old data versions) through queries that contain an appropriate time period specification.

Imagine that the row for Policy 1414 is deleted on 19 May 2011 with the following statement:

```
DELETE FROM policy WHERE id = 1414;
```

As shown in Table 13-9 and Table 13-10, DB2 removes the row from the current table and records the old version in the history table, setting the SYS_END column value for that row to the date of its deletion (19 May 2011).

Table 13-9 POLICY table current content after DELETE on 19 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	5000	N	250000	2011-05-19-13.30.46.194814065000	9999-12-30-00.00.00.000000000000

Table 13-10 Historical table content after DELETE on 19 May 2011

ID	VIN	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	10000	Y	500000	2011-05-18-22.56.06.249020907000	2011-05-19-13.23.19.707478331000
1111	A1111	10000	Y	750000	2011-05-19-13.23.19.707478331000	2011-05-19-13.30.46.194814065000
1414	B7777	14000	N	750000	2011-05-18-22.56.06.249020907000	2011-05-19-13.42.38.535783172000

Querying a system-period temporal table

Querying a system-period temporal table is simple. The syntax and semantics of basic SELECT statements remain unchanged. In particular, SELECT statements without any period specifications apply to data in the current table, as you might expect. Thus, existing applications, stored procedures, and database reports will not be impacted by adding system-period data versioning to your existing tables. Instead, you are simply able to include period specification as a means of specifying a point of time or time range in your SELECT statements to transparently access historical data (or a combination of current and historical data).

Let us explore a few examples so you can see how to write temporal queries on a system-period temporal table. But first, we start with the most basic scenario, a situation where you need to access only the most current information.

Accessing the current version of a row

Imagine that your current and history tables contain the car insurance policy data shown in Table 13-9 and Table 13-10. Then consider the following query:

```
SELECT coverage_amt FROM policy WHERE id = 1111;
```

As you might expect, DB2 returns one row with a coverage amount of 250000. (This is the amount stored in the row that contains current information about Policy 1111.)

Selecting with period specifications

What happens if you want to work with older versions of data? You simply include one of three supported period specifications in the FROM clause of your query:

- FOR SYSTEM_TIME AS OF ...

  This specification enables you to query your data as of a certain point in time.

- FOR SYSTEM_TIME FROM ... TO ...

  This specification enables you to query your data from a certain time to a certain time. DB2 uses an inclusive, exclusive approach for this period specification. In other words, the specified start time is included in the period but the specified end time is not.
- FOR SYSTEM_TIME BETWEEN ... AND ...

  This specification enables you to query your data between a range of start/end times. DB2 uses an inclusive, inclusive approach for this period specification. In other words, the specified start and end times are both included in the period.

Selecting information as of a point in time

In Example 13-5, we show a query to obtain information about the coverage amount recorded in the database for Policy 1111 at 1 a.m. on 19 May 2011.

Example 13-5 Coverage at a point in time

```
SELECT *
FROM R2.POLICY FOR SYSTEM_TIME AS OF TIMESTAMP('2011-05-19-01.00.00')
WHERE ID = 1111
```

To resolve this query, DB2 transparently accesses data in the history table to retrieve the correct information, specifically, to return a value of 500000 for this query. Note that you did not need to reference the history table in your query. The FOR SYSTEM_TIME period specification causes DB2 to automatically access the history table as appropriate.

Selecting from a point in time to current

In Example 13-6, we show a query to determine the total number of policy records for vehicle A1111 since 1:25 p.m. on 19 May 2011.

Example 13-6 Coverage over a period of time

```
SELECT COUNT(*) FROM R2.POLICY FOR SYSTEM_TIME FROM
TIMESTAMP('2011-05-19-13.25.00') TO TIMESTAMP('9999-12-30')
WHERE VIN = 'A1111'
```

Given our sample data, this query returns a value of 2. As shown in Table 13-9 on page 306 and Table 13-10 on page 306, one row in the current table and one row in the history table (the second row) meet the query's criteria. Note that this query references only the current table in the FROM clause, but DB2 automatically accesses the history table due to the system time period specification.

Selecting all available information for a policy

In Example 13-7, we show a query to see all the rows for a policy, including all available history.

Example 13-7 Select all available information

```
SELECT * FROM R8.POLICY FOR SYSTEM_TIME FROM TIMESTAMP('0001-01-01')
TO TIMESTAMP('9999-12-30') WHERE VIN='A1111' ORDER BY SYS_START DESC
```

Table 13-11 contains all information for vehicle A1111.

Table 13-11 All information for vehicle A1111

id	vin	annual_ mileage	rental_ car	coverage _amt	sys_ start	sys_ end
1111	A1111	5000	N	250000	2011-05-19-1 3.30.46.1948 14065000	9999-12-30-2 4.00.00.0000 00000000
1111	A1111	10000	Y	750000	2011-05-19-1 3.23.19.7074 78331000	2011-05-19-1 3.30.46.1948 14065000
1111	A1111	10000	Y	500000	2011-05-18-2 2.56.06.2490 20907000	2011-05-19-1 3.23.19.7074 78331000

13.1.3 Tracking effective dates with BUSINESS_TIME

The users, not the DB2 system, control what values are stored for the application period. The application period can be considered as a period showing when a row is valid to the user.

As we mentioned earlier, BUSINESS_TIME involves tracking when certain business conditions are, were, or will be valid. For example, a given product might have been priced at $45 during one month and $50 during another month. Or a credit card might have an interest rate of 16% one year and 18% the next year. BUSINESS_TIME is useful in such situations, because it enables applications to track and manage effective dates easily.

Like a system-period temporal table, an application-period temporal table requires the use of a time period, that is, the start and end points of the business condition. However, unlike a system-period temporal table, there is no separate history table. Past, present, and future effective dates and their associated business data are all maintained in a single table. In addition, users supply the start and end values for their business time period columns when they write data to the database. Finally, there is no need for a transaction-start-ID column.

Let us explore how to use this new DB2 technology in our sample application scenario.

Creating an application-period temporal table

Creating an application-period temporal table merely involves including appropriate columns for the start and end points of the time period and defines a BUSINESS_TIME period.

A period begin or end column must be defined as:

```
TIMESTAMP(6) WITHOUT TIME ZONE NOT NULL or DATE NOT NULL
```

Here is a defined BUSINESS_TIME period:

```
PERIOD BUSINESS_TIME(BUS_START,BUS_END)
```

Both period columns cannot be a user-defined distinct type.

An application period temporal table may have an index that is unique over a period of time, for example:

```
CREATE UNIQUE INDEX ix_policy_info ON policy_info (id, BUSINESS_TIME WITHOUT
OVERLAPS);
```

Here is a simple example that creates a table for car insurance policies, including their effective business dates. In Example 13-8, the BUS_START and BUS_END columns are defined as DATE data types. The PERIOD BUSINESS_TIME clause instructs DB2 to use these columns to track the start and end points of business time values for each row. To ensure temporal data integrity, DB2 automatically generates an implicit constraint to enforce that BUS_START values are smaller than BUS_END values.

Example 13-8 Creating the insurance policy table

```
CREATE TABLE policy (
id INT NOT NULL,
vin VARCHAR(10),
annual_mileage INT,
rental_car CHAR(1),
coverage_amt INT,
bus_start DATE NOT NULL,
bus_end DATE NOT NULL,
PERIOD BUSINESS_TIME(bus_start, bus_end),
PRIMARY KEY(id, BUSINESS_TIME WITHOUT OVERLAPS) );
```

The primary key constraint in this CREATE TABLE statement uses the optional clause BUSINESS_TIME WITHOUT OVERLAPS. This clause instructs DB2 to ensure that primary key values are unique for any point in business time. In terms of our insurance policy example, BUSINESS_TIME WITHOUT OVERLAPS means that there cannot be two *versions* or *states* of the same policy that are valid at the same time.

You can also use the ALTER TABLE statement to modify existing tables to track business time. To do so, you would need to add appropriate DATE or TIMESTAMP columns and define the period BUSINESS_TIME.

Inserting data into an application-period temporal table

Inserting a row into an application-period temporal table is straightforward: You simply need to supply appropriate values for all NOT NULL columns, including the columns representing business time start and end values. For example, to insert a few rows into our sample POLICY table with business time, we issue the statements shown in Example 13-9.

Example 13-9 Insert into POLICY table

```
INSERT INTO policy
VALUES(1111,'A1111',10000,'Y',500000,'2010-01-01','2011-01-01');
INSERT INTO policy
VALUES(1111,'A1111',10000,'Y',750000,'2011-01-01','9999-12-30');
INSERT INTO policy
VALUES(1414,'B7777',14000,'N',750000,'2008-05-01','2010-03-01');
INSERT INTO policy
VALUES(1414,'B7777',12000,'N',600000,'2010-03-01','2011-01-01');
```

In Table 13-12, we show the contents of the policy table after the inserts.

Table 13-12 POLICY table after INSERT

ID	VIN	annual_ mileage	rental_ car	coverage _amt	bus_ start	bus_ end
1111	A1111	10000	Y	500000	2010-01-01	2011-01-01
1111	A1111	10000	Y	750000	2011-01-01	9999-12-30

ID	VIN	annual_ mileage	rental_ car	coverage _amt	bus_ start	bus_ end
1414	B7777	14000	N	750000	2008-05-01	2010-03-01
1414	B7777	12000	N	600000	2010-03-01	2011-01-01

It may help to summarize the contents of this table in business terms. Very briefly, the data shows that Policy 1111 had a coverage amount of 500000 in effect from 1 January 2010 until 1 January 2011. From 1 January 2011 onwards, a coverage amount of 750000 is in effect. Similarly, the table shows that from 1 May 2008 to 1 March 2010, Policy 1414 had a coverage of 750000 for a specific vehicle with an estimated annual mileage of 14000. Effective 1 March 2010 to 1 January 2011, the coverage for this policy changed to 600000, and the insured vehicle was expected to be driven 12000 miles annually.

Let us consider the impact of the temporal uniqueness constraint we defined earlier on this table (with the BUSINESS_TIME WITHOUT OVERLAPS clause). Imagine that we issued the INSERT statement shown in Example 13-10.

Example 13-10 Another INSERT into the POLICY table

```
INSERT INTO policy
VALUES(1111,'A1111',10000,'Y',900000,'2010-06-01','2011-09-01');
```

DB2 would reject this statement and issue an error message because the statement attempts to add a row for Policy 1111 during the same time that one or more other rows are considered valid for this policy. This action would violate the temporal uniqueness constraint. If we intend to adjust the coverage of Policy 1111 from 1 June 2010 until 1 September 2011, we would accomplish this task using an appropriate UPDATE statement.

Updating data in an application-period temporal table

As you might imagine, you can still write traditional UPDATE statements for an application-period temporal table. In addition, you can also use the new FOR PORTION OF BUSINESS_TIME clause to restrict the update to a specific business time period. If your update impacts data in a row that is not fully contained within the time period you specified, DB2 updates the row range specified by the period clause and inserts additional rows to record the old values for the period not included in the update operation.

Let us review an example to see how this process works.

Imagine that you want to update information for Policy 1111 for a portion of time from 1 June 2010 to 1 September 2011, specifically, you want to alter the coverage amount for that period of time. In Example 13-11, we show the UPDATE statement.

Example 13-11 Update a policy

```
UPDATE policy
FOR PORTION OF BUSINESS_TIME FROM '2010-06-01' TO '2011-09-01'
SET coverage_amt = 900000
WHERE id = 1111;
```

Note that the temporal restriction in the query (FOR PORTION OF BUSINESS_TIME FROM . . . TO . . .) appears after the table name, not as part of the WHERE clause.

As shown in Table 13-12 on page 309, there were originally two rows for Policy 1111. Both rows are affected by our UPDATE statement, because the portion of business time that is being updated overlaps partially with the business period of each row. This overlap is illustrated in the upper part of Figure 13-1. When DB2 applies the UPDATE, each of the two original rows is split into two rows, as illustrated in the lower part of Figure 13-1. DB2 adjusts the effective dates of the rows automatically.

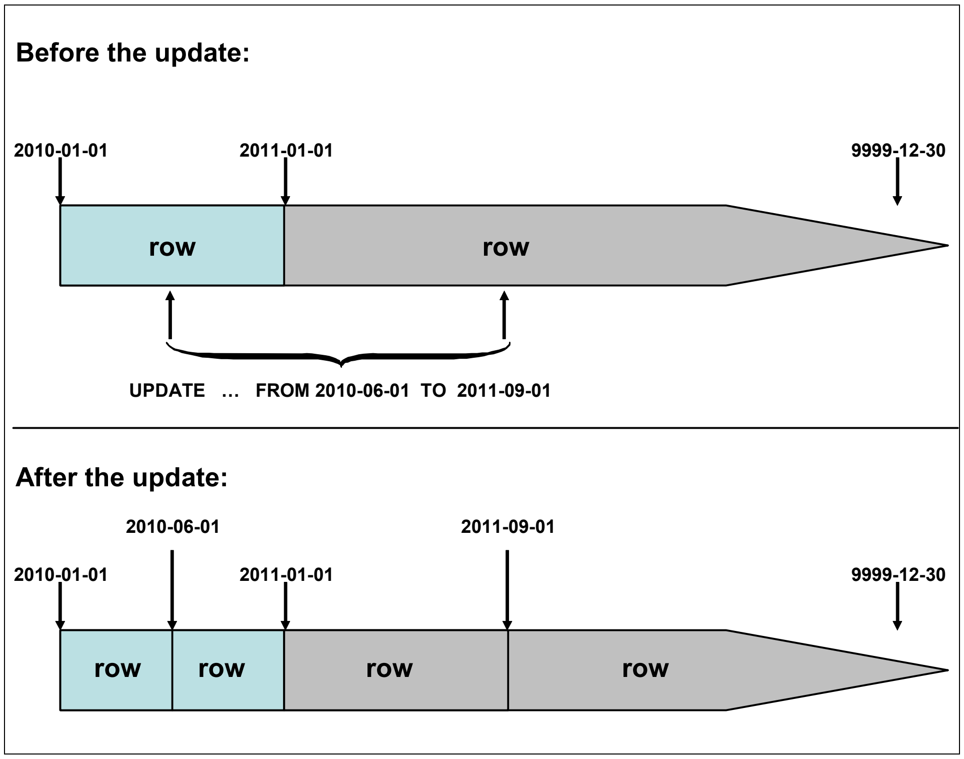


Figure 13-1 Row splits caused by the UPDATE statement

In Table 13-13, we show the resulting state of the POLICY table. The first row from Figure 13-1 is split into two new rows, shown in light gray in Table 13-13. The second row from Figure 13-1 also is split into the next two new rows, shown in dotted gray in Table 13-13.

Table 13-13 POLICY table after UPDATE of Policy 1111

ID	VIN	annual_ mileage	rental_ car	coverage _amt	bus_ start	bus_ end
1111	A1111	10000	Y	500000	2010-01-01	2010-06-01
1111	A1111	10000	Y	900000	2010-06-01	2011-01-01
1111	A1111	10000	Y	900000	2011-01-01	2011-09-01
1111	A1111	10000	Y	750000	2011-09-01	9999-12-30
1414	B7777	14000	N	750000	2008-05-01	2010-03-01
1414	B7777	12000	N	600000	2010-03-01	2011-01-01

Deleting data from an application-period temporal table

If you want to delete data from an application-period temporal table, you can restrict the delete operation to a specific range of time by specifying the FOR PORTION OF BUSINESS_TIME clause. If a row to be deleted has data that is not fully contained within the specified time range, DB2 ensures that the appropriate information from the row is preserved.

Imagine that a client wants to suspend his car insurance policy from 1 June 2010 to 1 January 2011. Assuming the client is referring to Policy 1414, the DELETE statement in Example 13-12 accomplishes this task.

Example 13-12 Suspend policy for a period of time

```
DELETE FROM policy
FOR PORTION OF BUSINESS_TIME FROM '2010-06-01' TO '2011-01-01'
WHERE id = 1414;
```

In Table 13-14, we illustrate the resulting contents of the table. Note that DB2 altered the final row (shown in gray) to reflect the new BUS_END date for Policy 1414.

Table 13-14 POLICY table after DELETE of Policy 1414

ID	VIN	annual_ mileage	rental_ car	coverage _amt	bus_ start	bus_ end
1111	A1111	10000	Y	500000	2010-01-01	2010-06-01
1111	A1111	10000	Y	900000	2010-06-01	2011-01-01
1111	A1111	10000	Y	900000	2011-01-01	2011-09-01
1111	A1111	10000	Y	750000	2011-09-01	9999-12-30
1414	B7777	14000	N	750000	2008-05-01	2010-03-01
1414	B7777	12000	N	600000	2010-03-01	2010-06-01

Querying data in an application-period temporal table

Querying data in an application-period temporal table is simple. Three optional period specifications enable you to specify temporal queries so that you can assess past, current, and future business conditions. Of course, you can still write basic SELECT statements (that is, non-temporal queries) against a table with a business time period, and DB2 processing of such queries will remain unchanged.

We explore a few examples so you can see how easy it is to write temporal queries involving business time. But first, we start with the most basic scenario, a situation where you do not need to consider any temporal conditions.

Imagine that your POLICY table contains the information shown in Table 13-15. (This is the same data shown in Table 13-12 on page 309, immediately after we created the POLICY table and inserted four new rows.)

Table 13-15 Sample contents of POLICY table

ID	VIN	annual_ mileage	rental_ car	coverage _amt	bus_ start	bus_ end
1111	A1111	10000	Y	500000	2010-01-01	2011-01-01
1111	A1111	10000	Y	750000	2011-01-01	9999-12-30

ID	VIN	annual_ mileage	rental_ car	coverage _amt	bus_ start	bus_ end
1414	B7777	14000	N	750000	2008-05-01	2010-03-01
1414	B7777	12000	N	600000	2010-03-01	2011-01-01

To determine the total number of insurance records that you have for Policy 1111, you could write the following query:

```
SELECT COUNT(*) FROM policy WHERE id = 1111;
```

Because this query contains no temporal predicates, DB2 returns a value of 2.

What if you want to consider various temporal conditions for these insurance policies? You simply include one of three supported period specifications in your query's FROM clause, right after the table name:

- `FOR BUSINESS_TIME AS OF ...`
- `FOR BUSINESS_TIME FROM ... TO ...`
- `FOR BUSINESS_TIME BETWEEN ... AND ...`

Let us step through an example. To obtain information about the coverage in effect for Policy 1111 on 1 December 2010, you could write the query shown in Example 13-13.

Example 13-13 Policy as of a point in business time

```
SELECT coverage_amt
FROM policy FOR BUSINESS_TIME AS OF '2010-12-01'
WHERE id = 1111;
```

DB2 returns a result of 500000.

To determine the terms applicable to Policy 1414 from 1 January 2009 until 1 January 2011, you could write the query shown in Example 13-14.

Example 13-14 Query results

```
SELECT *
FROM policy FOR BUSINESS_TIME FROM '2009-01-01' TO '2011-01-01'
WHERE id = 1414;
```

DB2 returns two rows, as shown in Table 13-16.

Table 13-16 Query results

ID	VIN	annual_ mileage	rental_ car	coverage _amt	bus_ start	bus_ end
1414	B7777	14000	N	750000	2008-05-01	2010-03-01
1414	B7777	12000	N	600000	2010-03-01	2010-06-01

13.1.4 Bitemporal tables

DB2 support for system and business time is straightforward. But the DB2 temporal data management capabilities do not end there. Indeed, we mentioned earlier that DB2 enables you to maintain both system and business time in bitemporal tables. For example, you may decide to use business time to manage your application's logical notion of time, such as the validity periods of insurance policies, and also use system time to track the history and time stamps of changes that transactions make to these policies.

Administrators can easily create or alter a table to include both system and business time. For example, the CREATE TABLE statement in Example 13-15 defines a bitemporal table with a BUSINESS_TIME period on the BUS_START and BUS_END columns and a SYSTEM_TIME period on the SYS_START and SYS_END columns.

Example 13-15 Bitemporal table

```
CREATE TABLE policy (
id INT NOT NULL,
vin VARCHAR(10),
annual_mileage INT,
rental_car CHAR(1),
coverage_amt INT,
bus_start DATE NOT NULL,
bus_end DATE NOT NULL,
sys_start TIMESTAMP(12) GENERATED ALWAYS AS ROW BEGIN NOT NULL,
sys_end TIMESTAMP(12) GENERATED ALWAYS AS ROW END NOT NULL,
trans_start TIMESTAMP(12) GENERATED ALWAYS
AS TRANSACTION START ID IMPLICITLY HIDDEN,
PERIOD SYSTEM_TIME (sys_start, sys_end),
PERIOD BUSINESS_TIME(bus_start, bus_end),
PRIMARY KEY(id, BUSINESS_TIME WITHOUT OVERLAPS)
);
```

After creating this bitemporal table, you need to create a compatible history table and enable versioning, as described in 13.1.2, “Creating a system-period temporal table” on page 301. Then you can insert, update, delete, and query rows in this table using the syntax described in “Querying a system-period temporal table” on page 306 and “Creating an application-period temporal table” on page 308.

13.2 Using temporal tables for auditing

We describe a scenario thank takes place in a bank, where a new mortgage application is being converted from VSAM to DB2 and is ready to be implemented into the production environment.

We first describe the general security set up in 13.2.1, “Basic security setup” on page 315 and then how an auditing requirement could be satisfied with a non-temporal scenario in the next four sections (13.2.1, “Basic security setup” on page 315, 13.2.2, “Production setup” on page 315, 13.2.3, “Interest rates scenario” on page 316, and 13.2.4, “Investigation” on page 317).

Starting with 13.2.5, “Altering the mortgage table to become aware of time” on page 321, we then show how DB2 temporal support could make the implementation of such scenario much simpler.

13.2.1 Basic security setup

A new CIO has recently joined the bank and one key area he is focusing on is security, that is, protecting the confidential data about the customers and the reputation of the company. The company has recently migrated to DB2 10, and many new options exist to improve security. One of the areas of concern is the availability of superuser IDs such as those holding the SYSADM authority; another is to move access control away from the system administrators and allow the smaller team of security administrators to manage access to the objects without having access to the data. These changes will be introduced gradually and the new SEPARATE_SECURITY DSNZPARM is set to NO for the first few months.

The production RACF group names and memberships are being re-evaluated, so when the mortgage application goes into production, authids PAOLOR8 and PAOLOR9 are granted the privileges to maintain the database, bind packages/plans, and run the nightly mortgage offline. This is temporary until a decision is made to use roles or RACF GROUPs. The current privileges available to PAOLOR8 and PAOLOR9 can be later easily revoked without cascade.

We show in Example 13-16 the granting of privileges. This action will be done by a person holding the SECADM authority who belongs to the new team of security administrators (RACF group=PROD_SEC). This group will eventually become the only one able to do such grants when SEPARATE_SECURITY is later set to YES.

Example 13-16 Granting of privileges

```
GRANT DBADM WITH DATAACCESS ON SYSTEM TO PAOLOR8,PAOLOR9;
---------+---------+---------+---------+---------+---------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------
GRANT BINDADD                         TO PAOLOR8,PAOLOR9;
---------+---------+---------+---------+---------+---------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

13.2.2 Production setup

PAOLOR9 creates the new mortgage tables and related objects. Example 13-17 shows the structure of the new mortgage table.

Example 13-17 MORTGAGE table definition

```
CREATE TABLE
   MTG.MORTGAGE
      (REF        DECIMAL (8)    NOT NULL,
       LOAN_AMT   DECIMAL(11,2)  NOT NULL,
       MAT        DATE           NOT NULL,
       REGION     CHAR(2)        NOT NULL,
       INTRATE    DECIMAL(5,3)   NOT NULL,
       COUNTRY    CHAR(3)        NOT NULL,
       PROPTYPE   DECIMAL(2,0)   NOT NULL,
       PRIMARY KEY (REF)
      ) IN DMTG.SMTGABC AUDIT NONE DATA CAPTURE CHANGES;
---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

After creating the table and populating the contents, PAOLOR9 realizes he forgot to set the AUDIT clause to CHANGES, which is the usual value for production. However, he remembers that audit policies can be created and dynamically activated. He asks one of the security administrators to establish an audit policy that is started every time DB2 comes up. PAOLOR1 is a member of the PROD_SEC RACF group and he inserts a new audit policy into SYSIBM.SYSAUDITPOLICIES for the mortgage table. The enforcement of this policy is totally independent of the current AUDIT option of the table. In Example 13-18, we show the INSERT statement into SYSIBM.SYSAUDITPOLICIES:

Example 13-18 Create audit policy on MORTGAGE table

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME,DB2START,OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
VALUES('MTGPOL','S','MTG','MORTGAGE','T','C');
---------+---------+---------+---------+---------+---------+---------+--
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

Note that when you specify DB2START='S', this policy will always be in effect at DB2 start (except for START DB2 LIGHT(YES)), and only users (authorization IDs or roles) with the SECADM authority can stop the policy.

In Example 13-19, we show how the security administrator activates the policy, so that the enforcement is immediate.

Here is the START TRACE command:

```
-STA TRACE (AUDIT) DEST (SMF) AUDTPLCY(MTGPOL)
```

Example 13-19 Activate the policy

```
DSNW130I  -DBOC AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSNW192I  -DBOC AUDIT POLICY SUMMARY
AUDIT POLICY MTGPOL STARTED
END AUDIT POLICY SUMMARY
DSN9022I  -DBOC DSNWVCM1 '-STA TRACE' NORMAL COMPLETION
```

PAOLOR1 then issues a DISPLAY TRACE command:

```
-DISPLAY TRACE(*) DETAIL(2)
```

In Example 13-20, we see the list of active traces. The list includes the MTGPOL audit policy trace.

Example 13-20 Display active traces

```
...
DSNW152I  -DBOC BEGIN TNO 03 QUALIFICATIONS:
NO QUALIFICATIONS
END TNO 03 QUALIFICATIONS
DSNW185I  -DBOC BEGIN TNO 03 AUDIT POLICIES:
ACTIVE AUDIT POLICY: MTGPOL
END TNO 03 AUDIT POLICIES
```

13.2.3 Interest rates scenario

In Table 13-17, we show the contents of one of the main mortgage tables prior to the weekend offline run.

Table 13-17 Initial contents of the MORTGAGE table

ref	loan_amt	mat	region	intrate	country	proptype
1111	130000.00	2019-01-31	CA	5.250	USA	1
2222	2520000.00	2017-10-31	MA	6.500	USA	2
3333	880000.00	2020-03-31	NV	4.150	USA	1
4444	450000.00	2018-06-30	WA	6.100	USA	3
5555	1150000.00	2019-01-31	CA	3.900	USA	2
6666	880000.00	2019-01-31	CA	7.550	USA	1

On Monday morning, one of the mortgage business managers runs an ad hoc report to see how the bank's mortgage interest rates compare with the current prime rate. The report looks unusual: The interest rates are all set to 99.99%.

The problem is immediately investigated. It looks like something happened between the Friday night offline run and the beginning of business on Monday morning. An auditor called Jim is also part of the investigation team. The first impression is that this is not a program bug but a malicious attack to affect the Bank's reputation and disrupt its ability to do business.

Jim tells the DBA (PAOLOR9) that he would like to see what the interest rates where prior to being maliciously changed.

The DBA mentions that it depends on the availability of information in the DB2 log and when such an update may have happened. He also mentions that the SMF file holding trace data is usually created at the end of the day where the entire day's information is split into files for CICS, for RACF and for DB2. Doing the investigation requires running a DB2PM report against the SMF files for the purpose of identifying suspicious transactions. The next step involves locating log archive files to run a DSN1LOGP utility to decipher the log records and establish what the rates were prior to any malicious activity. It also matters if the archive logs exist for the point in time of a malicious activity.

13.2.4 Investigation

PAOLOR9 knows that the audit policy was active for the MTG.MORTAGE table. The last good backup is from the end of Friday evening. He uses DB2PM to look through the SMF data set containing the DB2 data for Saturday. There is no unusual activity. Considering that the daily consolidated DB2 SMF data is a multistep process, there are opportunities to remove evidence of malicious activity.

The SMF data sets for Sunday have not yet been amalgamated and split into the files for RACF and CICS and DB2. PAOLOR9 needs to look through multiple raw SMF files for the Sunday period. There are 50+ SMF files to review.

PAOLOR9 runs the DB2PM job that covers one of the evening period. In Example 13-21, we show the input parameters for that period.

Example 13-21 Parameters for one of the evening periods

```
AUDIT
            TRACE FROM(05/22/11,19:50) TO(05/22/11,20:55)
                     DDNAME(AUDITDD)
EXEC
/*
```

Within the output, PAOLOR9 finds activity on the MORTGAGE table.The audit policy trace shows that there are six separate updates that are all associated with the same RBA.

We show the contents of the DB2PM report for the first statement in Example 13-22.

Example 13-22 Evidence of malicious activity

```
PACKAGE: DB0C.DSNESPCS.DSNESM68.X'149EEA901A79FE48'
TYPE: UPDATE                    STMT#   178  ISOLATION(CS)  KEEP UPD LOCKS:
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=99.99 WHERE REF=2222
DATABASE: DMTG              TABLE OBID:      3
ACCESS CTRL SCHEMA: N/P
ACCESS CTRL OBJECT: N/P

TYPE    : 1ST WRITE
DATABASE: DMTG                  TABLE OBID:      3
PAGESET : SMTGABC               LOG RBA   : X'00003OA3B22E'
```

The report also contains information about authid that ran the statement. In Example 13-23, we show more information associated with the update statement.

Example 13-23 The authid that ran the update statement

```
DB2R9    DB2R9    TSO          20:53:03.06 BIND
DB2R9    'BLANK'  C7CF3D3373C8
DSNESPCS TSO
```

So DB2R9 is the authid that made the update. We see in the OMEGAMON PE report that there are five other update statements against the MORTGAGE table. A total of 6 statements are recorded in the trace for the same unit of work.

Note that the ability to capture all the updates within a unit of work was not available prior to DB2 10. The AUDIT clause on a table only captures the first statement for a unit of work, and changing the AUDIT clause requires a REBIND. We use an audit policy and a trace command, which can capture more complete information and is not disruptive, as there is no REBIND. The audit policy trace can be turned on automatically when DB2 starts, which can improve security. You can further secure the audit policy by only allowing the security administrators holding the SECADM authority to stop the audit policy traces.

In Example 13-24, we consolidate the information and show all the statements that were run with authid DBR9. The single RBA associated with the statements is:

```
LOG RBA    : X'000030A3B22E'
```

Example 13-24   Consolidated statements

```
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=99.99 WHERE REF=1111
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=99.99 WHERE REF=2222
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=99.99 WHERE REF=3333
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=99.99 WHERE REF=4444
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=99.99 WHERE REF=5555
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=99.99 WHERE REF=6666
```

Now PAOLOR9 needs to determine the value of the interest rates prior to the update. He needs to run the DSN1LOGP utility against the archive log that covers that period of time. After much searching, PAOLOR9 finds a few files that may cover that time period.

We have the RBA associated with the time of the event, but to use DSN1LOGP, we need to know the DBID associated with the DMTG database and the OBID for the table space that contains the MTG.MORTGAGE table. We need these values in hexadecimal format. In Example 13-25, we display the SQL and results to accomplish this task.

Example 13-25   Get the DBID info

```
SELECT HEX(DBID) FROM SYSIBM.SYSTABLES WHERE NAME ='MORTGAGE';
---------+---------+---------+---------+---------+---------+---------+--

---------+---------+---------+---------+---------+---------+---------+--
014A
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

We obtain the corresponding OBID value from SYSIBM.SYSTABLESPACE. Now PAOLOR9 starts running the DSNLOGP utility against the log archive files that might contain the required information. In Example 13-26, we describe the parameters that are used.

Example 13-26   Parameters for DSN1LOGP

```
//SYSIN    DD  *
RBASTART (30A3B22E) RBAEND (30A3B22E)
DBID (14A) OBID(D)    SUMMARY(YES)
/*
//
```

In Example 13-27, we see activity against the table. We can see the value of the logical unit of work ID associated with the SQL statements that were used to make the updates.

Example 13-27   Using DSN1LOGP to find the LUWID

```
N1LPRT  UR  CONNID=TSO       CORRID=DB2R9        AUTHID=DB2R9     PLAN=DSNESPCS
  START DATE=11.142 TIME=16:53:03  DISP=INFLIGHT           INFO=COMPLETE
  STARTRBA=000030A3B22E  STARTLRSN=C7CF3D3379BE  NID=*
   LUWID=USIBMSC.SCPDBOC.C7CF3D3373C8.0001  COORDINATOR=*
   PARTICIPANTS=*
  NO DATA MODIFIED

 DATABASE WRITES PENDING (BASED ON INCOMPLETE LOG INFORMATION)
```

Now that we know the logical unit of work ID, we can run the utility again using the following parameter:

```
LUWID(USIBMSC.SCPDBOC.C7CF3D3373C8.0001) SUMMARY(NO)
```

In Example 13-28, we show the contents of the log undo redo record. It shows the change and the value of the interest rate prior to the malicious update.

Example 13-28 Content of the first log undo redo record

```
12I DSN1LGRD FIRST LOG  RBA ENCOUNTERED 000030956C59
...
0A3B2BE  TYPE( UNDO  REDO )  URID(000030A3B22E)
         LRSN(C7CF3D3379BE)  DBID(014A)  OBID(0002)  PAGE(00000002)      16:53:
         SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE  -  DATA CAPTURE)
         CLR(NO)  PROCNAME(DSNILREP)

*LRH* 006A0090 06000001 0E800000 30A3B22E 000030A3 B22E0726 000030A3 B22EC7CF  *
      3D3379BE 0000                                                            *
*LG** 88014A00 02000000 02000000 309EBF5C 2B02                                 *
0000  00320901 00032910 00170003 001F8000 F9999002 001F0003 01F00000 1111F000  *
0020  13000000 20190131 C3C1F052 50E4E2C1 F001                                 *
```

In Example 13-28, on the row below the *LG** row, we see the new value of the interest rate; it is F9999, which is 99.990%. On the row below, we also see the value of the interest rate prior to the update statement; this value is F052 50, which is 5.250%

Because the table was created with data capture changes, we also see additional information, such as the value of the REF column, which identifies the mortgage row; this value is 1111.

In Example 13-29, we display the information contained in the second undo redo record.

Example 13-29 Content of the second undo redo record

```
0A3B328  TYPE( UNDO  REDO )  URID(000030A3B22E)
         LRSN(C7CF3D337A83)  DBID(014A)  OBID(0002)  PAGE(00000002)      16:53:
         SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE  -  DATA CAPTURE)
         CLR(NO)  PROCNAME(DSNILREP)

*LRH* 006A006A 06000001 0E800000 30A3B22E 000030A3 B2BE0726 000030A3 B2BEC7CF  *
      3D337A83 0000                                                            *
*LG** 88014A00 02000000 02000000 30A3B2BE 2B00                                 *
0000  00320902 00032910 00170003 001F8000 F9999000 001F0003 02F00000 2222F002  *
0020  52000000 20171031 D4C1F065 00E4E2C1 F002                                 *
```

In Example 13-29, we use similar logic to deduce that the original rate for the mortgage with reference number 2222 was 6.500%.

We examine the remaining undo redo records associated with the same LUWID and consolidate the information as follows:

- Mortgage ref 1111 original interest rate was 5.250%.
- Mortgage ref 2222 original interest rate was 6.500%.
- Mortgage ref 3333 original interest rate was 4.150%.
- Mortgage ref 4444 original interest rate was 6.100%.

- Mortgage ref 5555 original interest rate was 3.900%.
- Mortgage ref 6666 original interest rate was 7.550%.

PAOLOR9 provides this information to Jim, the auditor. PAOLOR9 then does further analysis of any activity by DB2R9 since the last offline and none is apparent in the SMF files. After a discussion with Jim and other members of the investigation team, the consensus is to restore the mortgage database to the end of the last good offline run, which was Friday evening.

Coincidently, the person owning the DB2R9 user ID had been fired on the Thursday before the last offline. User ID DB2R9 had not yet been revoked from the system and it is likely that a disgruntled system administrator used that opportunity to cause some difficulties and loss of reputation for the bank.

The investigation was a lengthy and elaborate exercise. PAOLOR9 considers for a moment what would have happened if the archive logs were no longer available. Going forward, to avoid such difficulties, he decides to turn the mortgage table into a temporal table.

13.2.5 Altering the mortgage table to become aware of time

We can now implement the table level specifications to enable versioning of rows in the mortgage table and transform it into a system-period temporal table.

PAOLOR9 accomplishes this task in three high level steps as follows:

1. Issue the ALTER TABLE statement on the mortgage table to add row-begin and row-end columns, a transaction-start-ID column, and to add the system period.
2. Issue a CREATE TABLE statement to create a mortgage history table that will correspond to the system-period temporal table.
3. Issue the ALTER TABLE ADD VERSIONING statement with the USE HISTORY TABLE clause to define system-period data versioning on the table. This step establishes a link between the system-period temporal table and the history table.

First, in Table 13-18, we show the current contents of the mortgage table.

Table 13-18 Current contents of the MORTGAGE table

refq	loan_amt	mat	region	intrate	country	proptype
1111	130000.00	2019-01-31	CA	5.250	USA	1
2222	2520000.00	2017-10-31	MA	6.500	USA	2
3333	880000.00	2020-03-31	NV	4.150	USA	1
4444	450000.00	2018-06-30	WA	6.100	USA	3
5555	1150000.00	2019-01-31	CA	3.900	USA	2
6666	880000.00	2019-01-31	CA	7.550	USA	1

In Example 13-30, we alter the mortgage table to add three columns to keep track of time and define the system time period.

Example 13-30 Add columns and system period

```
ALTER TABLE MTG.MORTGAGE  ADD COLUMN SYS_START TIMESTAMP(12) NOT NULL
  GENERATED ALWAYS AS ROW BEGIN;
---------+---------+---------+---------+---------+---------+---------+--
```

```
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+--
  ALTER TABLE MTG.MORTGAGE  ADD COLUMN SYS_END  TIMESTAMP(12) NOT NULL
  GENERATED ALWAYS AS ROW END;
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+--
  ALTER TABLE MTG.MORTGAGE  ADD COLUMN TRANS_ID  TIMESTAMP(12) NOT NULL
  GENERATED ALWAYS AS TRANSACTION START ID;
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+--
  ALTER TABLE MTG.MORTGAGE
  ADD PERIOD SYSTEM_TIME(SYS_START, SYS_END);
---------+---------+---------+---------+---------+---------+---------+--
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

APAR PM31313 (currently open) is providing support for simplification and portability of the DDL for temporal tables. The DDL in Example 13-30 on page 321 could be written as shown in Example 13-31.

Example 13-31 Simplified ALTER sequence

```
ALTER TABLE MTG.MORTGAGE
   ADD COLUMN SYS_START TIMESTAMP(12) NOT NULL  GENERATED ALWAYS AS ROW BEGIN
   ADD COLUMN SYS_END TIMESTAMP(12) NOT NULL   GENERATED ALWAYS AS ROW END
   ADD COLUMN TRANS_ID TIMESTAMP(12) NOT NULL   GENERATED ALWAYS AS TRANSACTION
START ID
   ADD PERIOD SYSTEM_TIME(SYS_START, SYS_END);
```

In Example 13-32, we create the history table, which contains older versions of the current rows and versions of rows that are deleted. This table must be defined in a different table space than the mortgage table; it must also be the only table in this new table space. Note that when a table is identified in the LIKE clause and the table contains a row change time stamp column, a transaction-start-ID column, a row-begin column, or a row-end column, the corresponding column of the new table inherits only the data type of the original column. The new column is not considered a generated column.

Example 13-32 Creating the history table

```
CREATE TABLESPACE SMTGHST IN DMTG
   USING STOGROUP SYSDEFLT
     ERASE NO
   LOCKSIZE PAGE
   BUFFERPOOL BP2
   SEGSIZE 16
   CLOSE NO ;
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+--------

   CREATE TABLE MTG.MORTGAGE_HISTORY
           LIKE MTG.MORTGAGE
   IN DMTG.SMTGHST AUDIT NONE DATA CAPTURE CHANGES;
---------+---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

PAOLOR9 now enables system-period data versioning for the mortgage table. In Example 13-33, we show the alter statement that associates the mortgage table with the mortgage history table.

Example 13-33 Enabling data versioning

```
ALTER TABLE MTG.MORTGAGE
ADD VERSIONING USE HISTORY TABLE MTG.MORTGAGE_HISTORY;
---------+---------+---------+---------+---------+--------
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

13.2.6 New security measures

Jim, the auditor, is promoted and assigned new duties. Sally is the new auditor for the production DB2 environments. She has just started working at the bank and has much experience with open systems platforms. Sally is keen to monitor the use of trusted connections and the activities of system administrators that have superuser privileges. She puts in a request to the security administrator to establish and audit policy for this purpose.

In Example 13-34, we show the new audit policy being inserted into the audit policy table.

Example 13-34 New audit policy to monitor SYSADM and the use of trusted connections

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
           (AUDITPOLICYNAME,VALIDATE,SYSADMIN,DB2START)
 VALUES('SYSADM_POL','A','S','S');
---------+---------+---------+---------+---------+-------
DSNE615I NUMBER OF ROWS AFFECTED IS 1
```

The security administrator then starts the audit policy trace:

```
-STA TRACE (AUDIT) DEST (SMF) AUDTPLCY(SYSADM_POL)
```

In Example 13-35, we see that the trace starts.

Example 13-35 Trace started

```
DSNW130I  -DB0C AUDIT TRACE STARTED, ASSIGNED TRACE NUMBER 03
DSNW192I  -DB0C AUDIT POLICY SUMMARY
AUDIT POLICY SYSADM_POL STARTED
END AUDIT POLICY SUMMARY
DSN9022I  -DB0C DSNWVCM1 '-STA TRACE' NORMAL COMPLETION
***
```

Sally also determines that enough time has passed since the migration to DB2 10 and asks the security administrator to switch the DB2 subsystem to SEPARATE_SECURITY=YES.

Early on a long holiday weekend, the security administrator does an online DSNZPARM change to make the switch to separate security. The versioning feature and audit policies are now in place and SEPARATE_SECURITY is set to YES.

13.2.7 Next mortgage offline scenario

The next mortgage offline runs on the holiday on Monday in the evening, and the run is successful. Just after the offline occurs, we take a look at the contents of the mortgage table. In Example 13-36, we show the SQL that we use to display the active rows and their earlier versions.

Example 13-36 SQL for active and historical rows

```
SELECT * FROM MTG.MORTGAGE FOR SYSTEM_TIME FROM TIMESTAMP('0001-01-01')
TO TIMESTAMP('9999-12-30')
```

in Table 13-19, we see that the interest rates have changed during the mortgage offline run. Note that the rows coming from the history table are in the shaded cells.

Table 13-19 Active and historical rows after the offline run

ref	loan_ amt	mat	regi on	intrate	coun try	prop type	sys_start	sys_end	trans_id
1111	130000.00	2019-01-31	CA	5.750	USA	1	2011-05-23-21.19.45.059458519000	9999-12-30-00.00.00.000000000000	2011-05-23-21.19.45.059458519000
2222	2520000.00	2017-10-31	MA	6.700	USA	2	2011-05-23-21.19.45.059458519000	9999-12-30-00.00.00.000000000000	2011-05-23-21.19.45.059458519000
3333	880000.00	2020-03-31	NV	4.800	USA	1	2011-05-23-21.19.45.059458519000	9999-12-30-00.00.00.000000000000	2011-05-23-21.19.45.059458519000
4444	450000.00	2018-06-30	WA	6.350	USA	3	2011-05-23-21.19.45.059458519000	9999-12-30-00.00.00.000000000000	2011-05-23-21.19.45.059458519000
5555	1150000.00	2019-01-31	CA	4.200	USA	2	2011-05-23-21.19.45.059458519000	9999-12-30-00.00.00.000000000000	2011-05-23-21.19.45.059458519000
6666	880000.00	2019-01-31	CA	7.850	USA	1	2011-05-23-21.19.45.059458519000	9999-12-30-00.00.00.000000000000	2011-05-23-21.19.45.059458519000
1111	130000.00	2019-01-31	CA	5.250	USA	1	0001-01-01-00.00.00.000000000000	2011-05-23-21.19.45.059458519000	0001-01-01-00.00.00.000000000000
2222	2520000.00	2017-10-31	MA	6.500	USA	2	0001-01-01-00.00.00.000000000000	2011-05-23-21.19.45.059458519000	0001-01-01-00.00.00.000000000000
3333	880000.00	2020-03-31	NV	4.150	USA	1	0001-01-01-00.00.00.000000000000	2011-05-23-21.19.45.059458519000	0001-01-01-00.00.00.000000000000
4444	450000.00	2018-06-30	WA	6.100	USA	3	0001-01-01-00.00.00.000000000000	2011-05-23-21.19.45.059458519000	0001-01-01-00.00.00.000000000000

ref	loan_ amt	mat	regi on	intrate	coun try	prop type	sys_start	sys_end	trans_id
5555	1150000.00	2019-0 1-31	CA	3.900	USA	2	0001-01-01- 00.00.00.000 000000000	2011-05-23- 21.19.45.059 458519000	0001-01-01- 00.00.00.000 000000000
6666	880000.00	2019-0 1-31	CA	7.550	USA	1	0001-01-01- 00.00.00.000 000000000	2011-05-23- 21.19.45.059 458519000	0001-01-01- 00.00.00.000 000000000

We see that the interest values have changed during the last offline.

13.2.8 Problem with interest rates scenario

On the Tuesday morning, one of the mortgage business managers has a look at the trend in the bank's mortgage interest rates. As the economy is not doing so well, he is expecting to see all rates rising compared to last week. An ad hoc report shows that all rate changes were positive except for one, which decreased. The potential problem rate is for mortgage ref 2222. We run a query just for that mortgage, including the history rows, as shown in Example 13-37.

Example 13-37 Select for mortgage 2222

```
SELECT * FROM MTG.MORTGAGE FOR SYSTEM_TIME FROM TIMESTAMP('0001-01-01')
TO TIMESTAMP('9999-12-30') WHERE REF ='2222' ORDER BY SYS_START DESC;
```

In Table 13-20, we see all the rows related to mortgage 2222.

Table 13-20 Active and historical rows for mortgage 2222

ref	loan_amt	mat	regi on	intrate	coun try	prop type	sys_start	sys_end	trans_id
2222	2520000.00	2017-1 0-31	MA	6.050	USA	2	2011-05-23- 22.26.14.355 625253000	9999-12-30- 00.00.00.000 000000000	2011-05-23- 22.26.14.355 625253000
2222	2520000.00	2017-1 0-31	MA	6.700	USA	2	2011-05-23- 21.19.45.059 458519000	2011-05-23- 22.26.14.355 625253	2011-05-23- 21.19.45.059 458519000
2222	2520000.00	2017-1 0-31	MA	6.500	USA	2	0001-01-01- 00.00.00.000 000000000	2011-05-23- 21.19.45.059 458519000	0001-01-01- 00.00.00.000 000000000

We can see that the last change to this mortgage was made roughly an hour after the time of the changes made during the offline. For this mortgage, in the active row, the SYS_START value in Table 13-20 is about an hour after the SYS_START time in Table 13-19 on page 324.

An investigation is done, and using DB2PM, we determine that an SQL statement was run by PAOLOR9. We show in Example 13-38 the SQL statement information that was captured by running the audit policy trace.

Example 13-38 Update statement

```
TYPE: UPDATE                   STMT#   178  ISOLATION(CS)
TEXT: UPDATE MTG.MORTGAGE SET INTRATE=6.05 WHERE REF=2222
DATABASE: DMTG             TABLE OBID:      4
ACCESS CTRL SCHEMA: N/P
ACCESS CTRL OBJECT: N/P

TYPE    : 1ST WRITE
DATABASE: DMTG                 TABLE OBID:      4
PAGESET : SMTGABC              LOG RBA   : X'000030FBF7AC'
```

The person owning the PAOLOR9 authid is questioned and claims innocence. He is shocked and vehemently claims someone else must have been using his authid.

The bank investigators know that solid evidence is required to initiate a dismissal, so they take another look at the information in the DB2PM trace report; specifically, they look at the same section that contains the statement information shown in Example 13-38. To the left of that text information, we see additional relevant information. In Example 13-39, we show the primary auth and correlation name.

Example 13-39 Primary authid and correlation name

```
PRIMAUTH CORRNAME CONNTYPE
ORIGAUTH CORRNMBR INSTANCE
PLANNAME CONNECT               TIMESTAMP   TYPE
-------- -------- ------------ ----------- -------- ------------------
...

PAOLOR9  DB2R2    TSO         02:26:14.35 BIND     PACKAGE: DBOC.DSNESPCS.DSNE
PAOLOR9  'BLANK'  C7D0C98A181B                     TYPE: UPDATE
DSNESPCS TSO                                       TEXT: UPDATE MTG.MORTGAGE S
                                                   DATABASE: DMTG
                                                   ACCESS CTRL SCHEMA: N/P
                                                   ACCESS CTRL OBJECT: N/P
```

DBR2 is involved in this activity. We look at the DB2PM trace report. In Example 13-40, we see that a reuse of a trusted context took place one minute before the update statement. DB2R2 has SYSADM authority.

Example 13-40 Reuse of a trusted context

```
AUTHCHG  TYPE:              REUSE TRUSTED CONTEXT
         OBJECT OWNER:      AUTHID
         SECURITY LABEL:
         CONTEXT NAME:      TC_DB2R2_FOR_MAINTENANCE
         CONTEXT ROLE:
         USER ROLE:
         PREV. SYSAUTHID:   DB2R2
         REUSE AUTHID:      PAOLOR9
         SERVAUTH NAME:
         JOB NAME:
```

```
         ENCRYPTION:
         TCP/IP USED:
```

The investigators look into the catalog for the trusted context definition, shown in Example 13-41.

Example 13-41 Looking for the trusted context definition

```
SELECT NAME,DEFINER,SYSTEMAUTHID,CONTEXTID
         FROM SYSIBM.SYSCONTEXT WHERE SYSTEMAUTHID='DB2R2';
```

In Table 13-21, we see a trusted context defined for DB2R2.

Table 13-21 Trusted context definition

name	**definer**	**systemauthid**	**contextid**
TC_DB2R2_FOR_MAINTENANCE	DB2R2	DB2R2	72

Then another query is run against the catalog table that contains information about each authid with which the trusted context can be used. Example 13-42 shows the query.

Example 13-42 Who can use trusted context

```
SELECT NAME,DEFINER,SYSTEMAUTHID,CONTEXTID
         FROM SYSIBM.SYSCONTEXT WHERE SYSTEMAUTHID='DB2R2'
```

In Table 13-22, we see that PAOLOR9 is allowed to use the trusted context.

Table 13-22 Allow use for authid

contextid	**authid**
72	PAOLOR9

So we conclude that a trusted context was used by DB2R2 to take over the identity of PAOLOR9. The trusted context definition is shown in Example 13-43.

Example 13-43 Trusted context defined by DB2R2

```
CREATE TRUSTED CONTEXT TC_DB2R2_FOR_MAINTENANCE
BASED UPON CONNECTION USING SYSTEM AUTHID DB2R2
ATTRIBUTES (JOBNAME 'DB2R2')
WITH USE FOR PAOLOR9
ENABLE;
```

This trusted context was defined prior to the switch to SEPARATE_SECURITY=YES, when the people holding the SYSADM authority were able to define trusted contexts.

Prior to updating the rate, DB2R2 took over the identity of PAOLOR9 by changing the global DB2I DEFAULTS parameters before using SPUFI. Figure 13-2 shows the switch.

```
DB2I DEFAULTS PANEL 1
COMMAND ===>

Change defaults as desired:

 1  DB2 NAME ............. ===> DBOC      (Subsystem identifier)
 2  DB2 CONNECTION RETRIES ===> 0         (How many retries for DB2 connection)
 3  APPLICATION LANGUAGE   ===> IBMCOB    (ASM, C, CPP, IBMCOB, FORTRAN, PLI)
 4  LINES/PAGE OF LISTING  ===> 60        (A number from 5 to 999)
 5  MESSAGE LEVEL ........ ===> I         (Information, Warning, Error, Severe)
 6  SQL STRING DELIMITER   ===> DEFAULT   (DEFAULT, ' or ")
 7  DECIMAL POINT ........ ===> .         (. or ,)
 8  STOP IF RETURN CODE >= ===> 8         (Lowest terminating return code)
 9  NUMBER OF ROWS ....... ===> 20        (For ISPF Tables)
10  AS USER                ===> PAOLOR9   (Userid to associate with the trusted
                                           connection)
```

Figure 13-2 Switch to PAOLOR9

The person owning the DB2R2 authid was not aware that an audit policy was put in place to monitor SYSADM activity and to monitor the reuse of a trusted connections. The mortgage with reference number 2222 was for a property type 2, which is a shopping center. The person owning the DB2R2 authid was trying to help a friend. This friend was the owner of the shopping center and was having trouble making the monthly mortgage payments.

It is likely that DB2R2, after updating the rate, would have tried to alter the trusted context to remove its usage by PAOLOR9 by running a command, such as:

```
ALTER TRUSTED CONTEXT TC_DB2R2_FOR_MAINTENANCE DROP USE FOR PAOLOR9;
```

The above command would have resulted in a -552 error, as DB2R2 does not have the privileges to alter the trusted context.

DB2R2 thought that such a small rate change would go unnoticed and blend in with the normal changes done by PAOLOR9 while running the offline.

13.2.9 Conclusion

We saw in 13.2.4, “Investigation” on page 317 how onerous it was to research the before and after content of the data that was maliciously changed. Some shops only keep DB2 log archives for a couple of weeks, and when a request comes to investigate the effects of an SQL DML statement that ran three weeks ago, there may be limited information available. In our example, the changes were recent and the logs are still available.

It is likely that you do not need an audit trail for all tables in your application, but just a few critical ones. A system period temporal table can be easily established without breaking or changing the application and provide an audit trail of changes for the data you need to monitor.

You can now easily record all data changes, whether it is a change initiated by the customer (such as policy cancellation, beneficiary change, and so on) or the business (such as a insurance fee increase, policy rule change, and so on). Consider a banking example of an account merge event that needs to be investigated 6 months after it happened. Such an auditing functionality has been coded for years in many different ways and differently across applications in the same company. Up to 10 times reduction in the time to deployment has been observed when using DB2 10.

We showed in Example 13-37 on page 325 how easy it was to understand how the data changed. That change could have taken place months ago. Accessing this audit trail was simple and could be done by any user allowed to select from the table. Another prime benefit of this approach is the elimination of the risk of privileged user intervention, such as manipulation of SMF data and archived DB2 log data.

Using DB2 10:

- It is possible to implement a common approach to multiple applications, whether they are new or existing ones or home grown or purchased.
- Compliance can now be pushed away from the application into the database. A significant portion of compliance requirements may be met by preserving rows that are updated or deleted. History is now captured whether a table change is done within the application or outside the application, be it by SPUFI, batch, IBM QMF™, or a table editor.
- Powerful, novel, and intuitive SQL can immediately be used by business users and IT staff to query the same schema with new SQL extensions that are ISO standard. Such SQL can be used without IT Staff intervention.
- The response time for the current DML is preserved.
- Response time for past or future queries comparable.
- DBMS optimizer presents current and history tables as one, pulls answers from both as needed, and presents as one answer set.
- With application-period temporal tables, you not only can see the audit trail of the data changes made through the application, you now have a powerful feature that allows you to easily make corrections to data (past current or future) for a business time period.
- You can produce CPU reductions by using DB2 10 instead of using our own code. (Refer to *DB2 10 for z/OS Performance Topics*, SG24-7942 for more information about this topic.)
- Bitemporal tables can be used to see the data of a particular business “as of date” for a particular system time. These tables are useful in such industries as insurance and banking, for example.

The new DB2 10 temporal functionality provides great assistance in meeting requests to comply with regulatory needs, such as when an internal auditor declares that, going forward, your financial institution is now required to be able to report on changes made to a client’s records up to 5 years in the past.

Part 4

Security tools

In this part. we show details of implementation scenarios.

This part contains the following chapters:

14

Security tools for discovery and control

In this chapter, we introduce three tools that can be of assistance when providing a DB2 secure subsystem.

IBM Infosphere Discovery identifies and documents what data you have, where it is located, and how it is linked across systems by intelligently capturing relationships and determining applied transformations and business rules. It can help you identify critical or exposed data that can be the primary candidates for access control or data masking. Discovery can accelerate time to deployment for many IT projects, including:

- Data archiving
- Application retirement
- Application consolidation
- Master data management
- Sensitive data discovery and masking
- Data lineage discovery and documentation
- Data warehousing

IBM Tivoli Security Information and Event Manager leverages the capabilities of *IBM Tivoli zSecure* to enable inclusion of mainframe reporting on RACF into Tivoli Security Information and Event Manager enterprise-wide compliance dashboard and reporting, so that users can view the compliance status of their mainframe along with the rest of their environment. With these tools, you can centralizes log collection and event correlation across your whole enterprise.

SQL injection is considered one of the top and fastest growing web application vulnerabilities occurring in the database layer of an application. By using *pureQuery Runtime,* you can improve performance and, because the SQL is made available separately from the application, can also guard against SQL injection by restricting access to only SQL that has been vetted and approved. When the target database is DB2, the SQL can be bound into packages for static execution to improve performance, enhance security, improve manageability, and reduce costs.

This chapter contains the following topics:

- InfoSphere Discovery
- IBM Tivoli Security Solutions
- SQL injection and IBM Optim pureQuery Runtime

14.1 InfoSphere Discovery

A fundamental tenant of securing information is to provide the least level of authority or access required to perform a specific business task. To implement controls on sensitive data, and ensure that data access is provided within the appropriate “business need to know”, organizations must first decide what data constitutes sensitive information requiring protection. In some instances, this information is clearly spelled out within the different compliance initiatives in effect. In other situations, identifying the correct security classification becomes an institutional discussion, and can involve a number of different stakeholders within the organization. In many cases, data that might otherwise be viewed in an innocent context, and not classified as sensitive, might in fact represent critical metrics or contain proprietary information that, if exposed, can result in significant harm to an institution or its customers.

After the sensitive data elements are identified, and classified according to either internal or external compliance requirements, the next challenge is to locate which columns and tables contain this information. Many customers will look to their data dictionary repository for insights into answering this question, but in many situations, the information contained might not be reflective of their currently implemented database design. IBM has several different tools that can help organizations identify, classify, and categorize their information from a security standpoint.

For an ontime successful deployment of these critical initiatives, organizations must first possess a complete understanding of their existing data assets. Data discovery is the process of analyzing data values and data patterns to identify relationships. These disparate data elements are linked into logical units of information or “business objects”, such as customer, patient, or invoice. A business object represents a group of related attributes (columns and tables) of data from one or more applications, databases, or data sources. Discovery is also used to identify the transformation rules that have been applied to a source system when populating a target, such as a data warehouse or an operational data store. After these items are accurately defined, these business objects and transformation rules provide the essential input into information-centric projects, such as data integration, MDM, and archiving.

InfoSphere Discovery provides capabilities to automate the identification and definition of data relationships across complex, heterogeneous environments. Covering every kind of data relationship, from simple to complex, InfoSphere Discovery provides a 360 degree view of data assets. Standard relationships such as primary or foreign key constraints are defined directly in the database catalog. However, there are more complicated data relationships that are not easily visible. For example, application defined relationships are not contained in the database catalog, but instead are enforced through the processing logic of the application itself. In other situations, complex business rules might be applied to transform source data as it is moved into a target system.

InfoSphere Discovery analyzes the data values and patterns across heterogeneous sources to capture hidden correlations and bring them clearly into view. By applying heuristics and sophisticated algorithms, InfoSphere Discovery performs a full range of data analysis and transformation logic detection. It accommodates the widest range of enterprise data sources, including relational databases and any structured data source that can be represented in a text file format, such as a hierarchical database.

Without an automated process to identify data relationships and define business objects, organizations can spend months performing manual analysis, with no guarantee of completeness or accuracy. The InfoSphere Discovery automated capabilities accurately identify relationships and define business objects, speeding deployment of information-centric projects by as much as ten times. Faster implementation means that IBM clients can leverage new revenue and cost containment opportunities that are not available to the competition.

Some sensitive data is easy to find, for example, telephone numbers in a column named 'tel_number'. However, most application databases are more complex and sensitive data is sometimes compounded with other data elements or buried in text or comment fields. Data in one system may be moved and manipulated and used in other places. Subject matter experts (SME) can sometimes offer insight, but only if they fully understand the system. InfoSphere Discovery enables organizations to identify all instances of confidential data across their environment, whether clearly visible or obscured from view. InfoSphere Discovery works by examining data values across multiple sources to determine the complex rules and transformations that may hide sensitive content. It can locate confidential data items that are contained within larger fields, or that are separated across multiple columns, as shown in Figure 14-1.

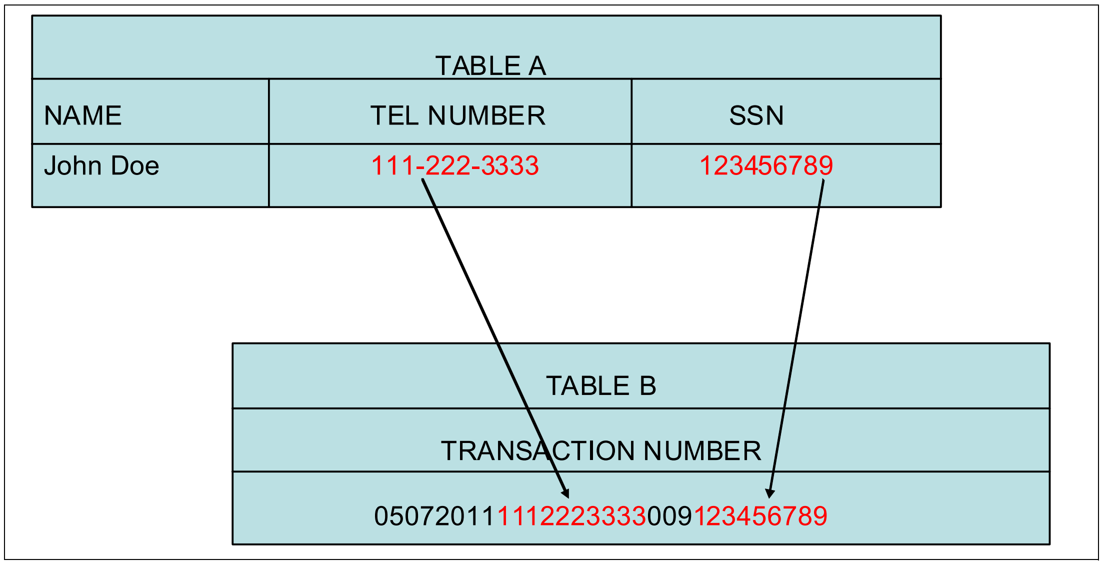


Figure 14-1 Embedded sensitive fields in related tables

InfoSphere Discovery delivers automated capabilities that offer greater accuracy and reliability than manual analysis. There is also the issue of disguising sensitive data, such as personally-identifying information, credit card numbers, or other confidential data contained in a database or data source. Most organizations recognize the need to mask or de-identify such data for purposes of compliance and good stewardship. Note that there are two aspects to these tasks:

- Discovering the sensitive data (either standalone, or embedded within text or comment fields or other unexpected places).
- What happens when you actually mask an identifier for security reasons.

In the latter case, you get similar issues as for data archival., that is, you need to maintain the relationship to sub-tables. Thus, if you use a social security number as a primary / foreign key and you do not de-identify both the parent and child columns consistently, you can makes orphans of all the data in the child table.

However, data stewards need to have an understanding of the data in a more general sense and this task, again, will be facilitated by the discovery of the ways in which different data elements are related to one another. Moreover, there is the whole question of compliance and periodic regulatory reporting, which will typically require an understanding of data relationships. We could include a separate heading for this task, but it certainly falls within the scope of data governance. A proactive approach to data governance would be to be able to prove that there is no data hidden away in unexpected places, such as email addresses or postal codes in a comments field, or credit card numbers in delivery instructions. This type of analysis could be based on example values, format, or domains.

InfoSphere Discovery provides capabilities to assist with many different data classification and discovery challenges. These capabilities include:

- Cross source column overlap analysis: Performs a cross compare of all the columns across many data sources to establish a baseline of overlapping data across multiple sources.
- Matching key prototype: Hypothesizes and tests the quality of matching keys on multiple data sources simultaneously.
- Empty target modeling and prototype: Drags, drops, and combines attributes from data sources to prototype a new unified schema. View the profiling statistics for the prototype target data.
- Precedence Discovery: Automatic generation of attribute matching precedence based on statistical analysis.
- Transformation Rule Discovery: InfoSphere Discovery features patented algorithms that automate the discovery of complex business rules between two structured data sets:
- Substrings, concatenations, cross-references, aggregations, case statements, arithmetic equations, and so on
- Automatic matching key discovery: Algorithms automatically discover the matching key and statistically validates the key between two data sources.
- Cross source data preview: Provides a side by side preview of data across multiple data sources for the same logical row and allows the analyst to see values that match the business rules and anomalies that do not match.
- Identification of sensitive information: Workflow supports classification of Personally Identifiable Information (PII).
- Business object creation: Defines complete business objects (logical groupings of related objects, for example, customer data) that serve as essential inputs into information-centric projects such as data integration, master data management, data warehousing test data management, and data archiving using Optim products.
- Import/Export: Reads mapping specs from CSV files and generates source maps to CSV files.
- Standardize business terms: Creates and manages your business vocabulary within IBM InfoSphere Business Glossary.

When used in conjunction with the IBM Optim Data Privacy Solution, InfoSphere Discovery provides the most effective enterprise-scale approach for locating and masking sensitive data across complex, heterogeneous environments.

InfoSphere Discovery V4.5 adds the following capabilities:

- Extended language support, allowing analysis and discovery of source data in Chinese and other Unicode character sets

- New asset classification workflow that applies advanced algorithms for improved value matching and classification
- Improvements to the user interface for interacting with large data sets
- Scalability and performance enhancements for working with larger databases and more tables during source analysis
- Integration with InfoSphere Business Glossary for capturing and defining terminology during the discovery and analysis process

14.2 IBM Tivoli Security Solutions

An architecture also must support many communities and represent the long-term view of a technical direction. IT security compliance architectures, in particular, must allow for multiple implementations depending on the *realities of the moment*. An organization must exercise caution to prevent the IT security compliance architecture from becoming a blueprint for a specific implementation.

IBM Tivoli zSecure Audit and Tivoli Security Information Event Manager (TSIEM) help provide a framework for IT security compliance functionality throughout the organization. An IT security compliance architecture must be flexible and open to deal with the changing environment that an organization might face in the future.

Security Information and Event Management (SIEM) helps an organization gather security data from many divergent information systems. The volume of security log data is growing over time with more and more systems being connected to an organization's infrastructure. Having all this information in a centralized storage helps an organization to better analyze the data and respond to auditors' requests during reviews and audits.

Many organizations are struggling with three major problems that they cannot completely, or even partially, fulfill:

- Demonstrating compliance to regulatory requirements
- Ensuring appropriate protection of intellectual capital and privacy information
- Managing security operations securely and effectively

An SIEM system can collect data from log files and alerts from a variety of infrastructure components, such as firewalls, routers, antivirus systems, servers, and many other components. It can inform IT teams about unusual behavior on these systems, and then these teams can decide whether and what kind of further investigation to take.

An SIEM architecture can be broken down into two elements:

- Security Information Management (SIM)

  The SIM component provides reporting and analysis of data primarily from host systems and applications and secondarily from security devices to support regulatory compliance initiatives, internal threat management, and security policy compliance management. It can be used to support the activities of the IT security, internal audit, and compliance organizations.

- Security Event Management (SEM)

  The SEM component improves security incident response capabilities. It processes near-real-time data from security devices, network devices, and systems to provide near-real-time event management for security operations. It helps IT security operations personnel be more effective in responding to external and internal threats.

An SIEM solution must provide log data capturing capabilities. Aggregated information must be securely stored. Also, archived data faces the requirement of having to reside in a database format that allows for accurate and expedient reporting and viewing capabilities.

14.2.1 Tivoli Security Information and Event Manager

An SIM solution requires log management and audit functionality and must generate reports on collected log data that refers to security policies to identify policy violations.

In addition, the solution must offer real-time incident management, which can require using correlation rules to identify more details about the incident. Ideally, the solution must offer real-time mitigation for and alerting of incidents, but strictly speaking, this type of real-time functionality is typically addressed in an SEM realm.

A combination of the Tivoli Security Information and Event Manager and Tivoli Security Operations Manager products can meet all the requirements in which Tivoli Security Operations Manager adds real-time incident management to the Tivoli Security Information and Event Manager solution. In addition, the Tivoli Security Information and Event Manager solution offers other capabilities that are often key decision factors for organizations that are looking for these types of technologies. Among these additional factors are a rapid and scalable deployment and support process.

The SIEM solution is modular and flexible and includes the capability to start an implementation out small to address key aspects of SIEM requirements and then grow to encompass the complete SIEM requirements of an organization at a later time.

The SIEM implementation can start to address the log management functionality first and address the other SIEM functions over time, in phases. Some times organizations that deploy SIEM solutions place emphasis in their evaluations on their vendor providing an appliance type solution.

The Tivoli Security Information and Event Manager Log Management solution provides the ability to collect log data in its original format using a FIPS-certified secure protocol. The original log data is stored in a secure, reliable, and verifiable way.

Verification of the log collection occurs on two levels in Tivoli Security Information and Event Manager 2.0:

- Verification of successful archiving
- Verification of log completeness

Regulatory compliancy requires that collected log data be archived. Tivoli Security Information and Event Manager Log Management produces reports to prove that the required log data was indeed archived successfully. Besides this proof, the Log Management component can also demonstrate that the archived log data is complete.

Tivoli Security Information and Event Manager Log Management includes the capability to perform content searches across the entire Log Management Depot when it is important to find critical information in the raw logs. Original log data analysis is possible in two ways:

- Forensic search: The original log data can be analyzed by running user-defined search queries. Like in a SQL query, the user defines the search criteria and the filed types that must be displayed from the result set of log records
- Best practice log analysis reports: Tivoli Security Information and Event Manager has built-in log analysis best practice reports that require minimal configuration.

Real-time event correlation and alerting

If there is a need for real-time event correlation and alerting, you must integrate Tivoli Security Information and Event Manager with Tivoli Security Operations Manager. Alerts that are identified by either of the products can be exchanged between them. Tivoli Security Information and Event Manager can be informed about both identification and mitigation of threats by Tivoli Security Operations Manager. This information sharing allows Tivoli Security Information and Event Manager to generate the proof through compliance reports that incident management is effective and also archive the evidence that supports the claim.

The Tivoli Security Operations Manager architecture uses a multi-phased approach to event correlation and analysis called *deterministic threat analysis*. This type of threat analysis gives a security team the ability to detect known and unknown attacks, internal misuse, misconfigurations, and other anomalous activity. Using deterministic threat analysis, every event goes through both computational (policy based) and rule-based correlation, which establishes a base line from which any change is detected.

Computational correlation involves using algorithms to assign values to events or groups of events based on factors, such as the severity of the event or the weighting that is assigned to a particular asset. These values are derived from your security policies and standards.

The Tivoli Security Operations Manager architecture uses two distinct techniques to perform computational correlation on events gathered by the system:

- *Impact correlation* relates to the calculations performed on individual events.
- *Statistical correlation* determines the trends that are created by groups of events.

Together, these two methods of calculating threats provide a highly accurate and surprisingly easy to administer method of prioritizing incidents of your network infrastructure without needing to configure rules.

The Tivoli Security Operations Manager architecture makes it possible to consolidate information from the network or insider attack that compromised the portal with the out of policy database changes. Analysts that use Tivoli Security Operations Manager can see the effects of an attack on applications and data and details of when the attack occurred, what user identities were used, and how it was accomplished. Using the comprehensive graphical interface, a non-technical user can focus on a specific event that occurred on the database. They can extend their analysis to view all activity by a particular user around the time of that event, including activity on other systems. They can drill down to the details about a specific event or database operation and look at all events on the database or events for a user on other systems around the time of a specific policy violation or alert. They can open reports, such as one on the actions of system administrators, to learn how a user was granted privileges and who granted them. These capabilities allow auditors and administrators to determine not only which user performed an action on a database, device, mail server, or other infrastructure component, but the actions that were performed by privileged users to grant that user the privileges that they need to perform actions that violated policy.

The Tivoli Security Operations Manager component adds real-time, rule-base threat management to the Tivoli Security Information and Event Manager solution. This component provides the following correlation techniques:

- Rule-based correlation
- Vulnerability correlation
- Statistical correlation
- Susceptibility correlation

Identified threats and their possible automatic mitigation can be picked up as events by Tivoli Security Information and Event Manager for integration into compliancy reporting.

With Tivoli Security Operations Manager added to it, Tivoli Security Information and Event Manager can cover the complete SIEM requirement spectrum from real-time threat and log management to regulatory and security policy compliancy reporting.

14.2.2 InfoSphere Guardium and TSIEM

InfoSphere Guardium can provide a robust solution for safeguarding your entire application and database infrastructure, which includes:

- Real-time *database activity monitoring* (DAM) to proactively identify unauthorized or suspicious activities
- Auditing and compliance solutions for simplifying SOX, PCI DSS, and data privacy processes
- Change control solutions for preventing unauthorized changes to database structures, data values, privileges, and configurations
- Vulnerability management solutions for identifying and resolving vulnerabilities
- Database leak prevention for locating sensitive data and thwarting data center breaches

InfoSphere Guardium provides the most widely-used solution for ensuring the integrity of corporate information and preventing information leaks from the data center.

The enterprise security platform can prevent unauthorized or suspicious activities by privileged insiders, potential hackers, and users of enterprise applications, such as Oracle EBS, PeopleSoft, SAP, Business Intelligence, and other in-house systems.

At the same time, the solution optimizes operational efficiency with a scalable, multi-tier architecture that can automate and centralize compliance controls across the entire application and database infrastructure.

The combination of TSIEM and InfoSphere Guardium for database activity monitoring provides a powerful and robust framework for information security. Using InfoSphere Guardium, real-time policy based activity monitoring can be implemented to detect suspicious activity at the database layer. As database events are collected, they are evaluated against policies for detection of inappropriate or malicious access. If a single or series of events violate an existing policy, InfoSphere Guardium can format an alert event, and immediately publish this event to Tivoli TSIEM for distribution and action. This interface between TSIEM and InfoSphere is through a predefined, industry standard interface.

14.3 SQL injection and IBM Optim pureQuery Runtime

In this section, we provide a description of SQL injection and show how Optim pureQuery Runtime can help remediate exposures.

14.3.1 SQL injection

With the wealth of new web applications replacing traditional transaction oriented applications such as CICS or IMS server-side applications, security must be designed into the system from the start. A common security hole in web-based applications is known as SQL injection.

SQL injection is a technique which enables an attacker to execute unauthorized SQL commands by taking advantage of non-validated input fields in applications that build dynamic SQL queries. This action typically occurs when the web application combines the strings of a query with an unvalidated input variable so that someone could add a second SQL statement or otherwise change the query to give them information or access that they should not have.

Structured Query Language (SQL) is an interpreted database language that is commonly used by applications to communicate to back-end database servers of web-based applications. The databases are where information is stored in records, also known as *rows*. A row might contain fields, or more accurately, columns, which can consist of a customer name, address, email address, phone number, and other information. This class of information is commonly referred to personally identifying information (PII), and the protection of PII is the subject of most of the common regulatory compliance initiatives in use today.

To view a row of customer data in a database, an SQL query is constructed by the application, and sent to the database server, which should return that customer's information and no other information. The SQL statement for this query could look like what is shown in Example 14-1.

Example 14-1 Sample SQL select

```
SELECT * FROM EMPLOYEES WHERE LAST_NAME='SMITH'</
```

In subapplication servers, the customer name is generally represented as a variable with a value that enters the application through a web form among data that comes from the application's user. So, in Java, the query might be constructed something like Example 14-2.

Example 14-2 Java example of sample SQL select

```
String query = "Select * from employees where last_name='" +
req.getParameter("lastName") + "'";</
```

In the normal functional case, a user enters a last name such as “Smith” into that field, but an attacker could just as easily enter a value of “' or '1'='1'-”. That sort of value would result in a query like the one shown in Example 14-3 being sent to the database server.

Example 14-3 Malformed SQL statement

```
SELECT * FROM EMPLOYEES WHERE LAST_NAME='' OR '1'='1'</
```

That Boolean expression (last_name='' or '1'='1') is always going to be true, which will result in the database server responding with the entire contents of the employee table. That is, our attacker has likely just succeeded in extracting the employee data of every one of our employees.

The relevant issue in the above scenario is that the intent of the original SQL statement can be altered by malformed user data. But Java and most web-related languages provide a simple solution called parameterized queries.

In a parameterized query, the data is passed to the SQL query as actual data, not as part of the SQL syntax itself. It takes only a little bit more effort to set up the query, but then it is done, and any data provided by an attacker will do nothing to alter the intent of our query.

Here is what our query would look like if we used a parameterized query. To avoid SQL injection with DB2, you should use prepared statements with parameter markers, as shown in Example 14-4.

Example 14-4 Java parameterized query

```
SELECT LAST NAME, HIRE_DATE FROM EMPLOYEES WHERE LAST NAME = ?
```

Each ? in the prepared statement is then replaced by exactly one variable that you either bind (using something like db2_bind_param()) or pass as an array to the execute() function. The execute() function will bind that input value to exactly one parameter (so that last name really is compared against “1' OR 1 = 1" rather than just 1 with the additional, always true OR clause appended to the statement).

Tip: As a best practice, code your applications with security in mind. Avoid retrieving redundant data and scrub all input values.

SQL injection attacks are mainly based on exploiting code not written with security in mind. To prevent these exploits, you could:

- Never trust any kind of input.
- Never connect to the database as a user with super authority or as the database owner. Use always connect to the database with credentials with limited privileges.
- Check if the given input has the expected data type.
- Quote each non-numeric user supplied value that is passed to the database with a database-specific parameter markers.
- Do not print out any database specific information, especially about the schema. Carefully reconsider the common practice of granting select privileges to public on the DB2 catalog objects.

Literals and SQL performance

The benefits of using parameters in the queries instead of literal values has been a topic of discussion for DB2 database administrators and developers for years. The main point of discussion has traditionally been better application SQL performance. In general, DB2 will perform much better and will use less server resources processing SQL statements that have parameters instead of queries that have literal values in them instead.

DB2 compiles (prepares) an SQL statement once and will cache it so that the next time it sees the same statement, it will not have to waste time and resources compiling / preparing the same statement, and instead will reuse the same query execution path from the cache. The problem is that when SQL statements use literal values instead of parameters, matching incoming statements to what is already in the statement cache becomes a much more difficult task for the database to do find a good existing match. Consider the simple SQL shown in Example 14-5 that uses a literal value for last name.

Example 14-5 Sample select with literal - 1

```
SELECT * FROM EMPLOYEE WHERE LASTNAME='MANCILL'
```

After the database sees this SQL statement, it compiles it and saves it in the statement cache. Let us say a query for a second employee is submitted, as shown in Example 14-6.

Example 14-6 Sample select with literal - 2

```
SELECT * FROM EMPLOYEE WHERE LASTNAME='ANTONELLI'
```

Because the queries do not look exactly the same, the cache matching algorithm will have a hard time understanding that this is exactly the same query and in many cases will re-prepare it again and will store it in the statement cache as well. As a result:

- DB2 will keep compiling the same statement over and over again.
- DB2 wastes memory to hold more and more copies of the same statement in the cache.

Now, let us assume that your application developer makes a simple change and codes a parameterized SQL query, as shown in Example 14-7.

Example 14-7 Parameterized SQL statement

```
SELECT * FROM EMPLOYEE WHERE LASTNAME = ?
```

DB2 could cache the statements and avoid the penalty associated with using literal values.

Security model when executing SQL statements statically

In the SQL static execution model, the authorization ID that is used at run time is not required to have access to base database objects. Instead of giving access to base database objects as in a dynamic JDBC implementation, the runtime authorization ID is given access to a specific predefined package and the included SQL statements.

Authorization of a package allows an improved security implementation because authorization IDs cannot change the SQL statements by using programming logic. In addition, if the ID is used to connect from an alternate access mechanism due to a security breach, the ID cannot execute SQL dynamically. Authorization of a package also allows a strict auditing of all SQL statements that will be executed against a set of tables.

In the static model, a user who has the authority to access the base table will bind the packages and become the package owner. The package owner can then grant the ability to execute the package to a runtime environment authorization ID, such as the user ID stored within the WebSphere Application Server data source definition. The WebSphere user ID cannot execute any other SQL dynamically against the database objects outside of the SQL statements that are defined in the package.

In contrast, in a dynamic SQL environment, a user ID is used for all data access on behalf of the users connected to the application server. This user ID can execute SQL dynamically against the database outside of the application server environment.

14.3.2 Optim pureQuery Runtime for z/OS for protection from SQL injection

Optim pureQuery Runtime has an SQL literal substitution capability that can convert non-parameterized SQL statements that the application executes into parameterized SQL statements. This capability is useful for applications that generate SQL statements dynamically during run time. Additionally, for SQL statements that execute dynamically, SQL literal substitution reduces the total number of SQL statements in the server dynamic cache, which increases the cache-hit rate for statements.

To avoid the issue of SQL injection, where an application's data access layer was not developed according to best practices, you can use the pureQuery Runtime's SQL literal substitution feature in combination with its whitelist processing to provide protection. During the capture phase, SQL is consolidated into a single statement if it matches exactly (with the exception of the string literal values). This consolidated SQL statement is stored in the SQL capture file and matched against incoming SQL at run time. If, at run time, an incoming SQL statement can be reduced to the parameterized statement that is present in the capture file, the parameterized SQL will be executed instead and the SQL literal passed through the database as a typed parameter value.

In the case of a malicious user, however, the incoming SQL statement in the example above cannot be matched because it contains more SQL literals. Because no matching occurred, the query is executed as though whitelist processing is not enabled and an unauthorized data access occurs. The Optim pureQuery Runtime whitelist processing feature prevents this unauthorized execution, because this form of SQL (with additional SQL literals) is not present in the approved whitelist. This runtime functionality essentially mirrors what happens during static SQL execution without needing to go through a client optimization process. When an application is enabled with pureQuery client optimization, you can specify which of the SQL statements that are executed by application are executed statically and which are executed dynamically. For example, you can choose to run the most critical SQL statements statically and allow others to execute dynamically.

Developers do not need to know or understand details of the database artifacts that affect static SQL execution or the deployment steps that are needed to convert the application to execute SQL statically. Developers can develop applications that execute SQL dynamically based on the framework with which they are familiar. Developers can focus on functional characteristics of the application and the business logic of the application. The database administrators can focus on deploying and optimizing the SQL statements that are used by the application.

When executing SQL statically, pureQuery client optimization can use the package versioning capability of static SQL in a DB2 database server. Static SQL execution allows a staged rollout with concurrent execution of new application code and DB2 access plan changes. Package versioning lets you save existing DB2 packages and create new packages. You can use package versions to create a clear and simple fallback procedure in cases when the changes have negative effects.

In JDBC-based dynamic SQL applications, no fallback procedure exists for DB2 access plan changes. When executing SQL statically, you can use package versioning to execute SQL with a previously generated access plan. Using pureQuery client optimization for an existing JDBC application makes it easy to substitute executed SQL with an optimized equivalent SQL. A database administrator can replace an optimized SQL statement for a statement that is performing poorly without modifying the application.

The replaced SQL statement can contain changes like additional WHERE clauses for better index selectivity. The replaced SQL statement cannot add parameters or result columns because the metadata is already captured.

The pureQuery platform does not replace the database's security framework, but it can be used to augment it. Using the client optimization process, Java data access applications that were implemented using the JDBC API can be enabled to take advantage of the static SQL security model. Java applications where the data access layer was implemented using the pureQuery method style API natively support static SQL execution and can therefore use this security model. If dynamic SQL execution is desired, the Optim pureQuery Runtime can be used to restrict which SQL can be executed by a particular application.

For a technical description of this tool, refer to *IBM Optim pureQuery Runtime for z/OS performance,* available at the following address:

http://www.ibm.com/developerworks/data/library/dmmag/DBMag_2008_Issue2/pureQueryRuntime/index.html

15

Auditing and InfoSphere Guardium

In this chapter, we introduce the InfoSphere Guardium database activity monitoring solution. InfoSphere Guardium provides a simple, robust solution for ensuring the privacy and integrity of trusted information by automating the entire compliance auditing process in heterogeneous environments.

We also show some additional features to provide for a database vulnerability assessment approach, as well as how to discover where sensitive data is located.

We have added this information on auditing and InfoSphere Guardium here because DB2 data is often only a portion of the critical data of the enterprise and users often need a framework for real-time database activity monitoring.

This chapter contains the following topics:

- InfoSphere Guardium
- Database security functionality using InfoSphere Guardium
- InfoSphere Guardium S-TAP for DB2 for z/OS

15.1 InfoSphere Guardium

As we described in 12.1, “Activity monitoring options on DB2 for z/OS” on page 276, the combination of collecting audit event records with audit policy, processing the resultant events into DB2 tables, and writing SQL based audit based processes allows you to create a relevant audit trail report. However, there are some concerns with audit and activity monitoring based just on the native audit trace facility.

One element of any auditing approach that uses the native trace facility is that DBADM/SYSADM can interfere with or stop the collection of audit events by issuing the STOP TRACE command. DB2 10 for z/OS audit policies provide much better audit event collection capabilities and also provide the ability to prevent traces from being stopped unless they are stopped by SECADM. However, there still are some issues in externalizing the data.

Because the audit events are ultimately externalized through IFCID collection to SMF, this collection data can still be interfered with or modified. The z/OS service aid utility program IFASMFDMP can be used to process the SMF data sets and selectively exclude or remove events from the collected SMF data sets. In addition, the IFCID 143/144 would not include the host variable information for any SQL statement using parameter marker processing.

Because DB2 10 audit policy collects more event data with the removal of the first SQL statement in each logical unit or work restriction, you can speculate that the impact will increase as more events are generated. Also, remember that once the data has been collected by DB2, it is then externalized to SMF, and the downstream processing needs to be performed, all of which results in additional z/OS CPU resource consumption. The current performance objective with InfoSphere Guardium S-TAP for z/OS is single digit impact, depending on audit objectives and customer SQL workload mix, and in some cases will be in the mid range of this single digit number. Also, after the SQL event is collected by InfoSphere Guardium S-TAP for z/OS and streamed to the InfoSphere Guardium appliance infrastructure, no additional z/OS host CPU is consumed and the opportunity for interference is minimized.

More importantly, while an audit policy makes the collection of the audit event information more granular, and provides for better administration protection around the starting and stopping of the trace collection, it does not provide any reporting capability. In the absence of a robust reporting tool, such as Guardium, it is incumbent on the customer to turn the audit event data collected with DB2 10 audit policy into meaningful reports. All of this translates into additional z/OS CPU cycles to generate audit report information, increased complexity, and reliance on a *home grown* solution for audit reporting, and problematic separation of roles and responsibilities.

One other fundamental fact is that database activity reporting is essentially an *after the fact* exercise. As previously mentioned, in the case of reporting based on audit event data generated from the DB2 10 for z/OS audit policy, data is typically externalized to SMF. In many implementations, there can be significant latency between the audit event being collected through the IFCID process, and the audit report being generated and reviewed by security personnel. With InfoSphere Guardium for z/OS, as events are collected, they are immediately streamed through TCP/IP to the InfoSphere Guardium appliance infrastructure, and are immediately available for reporting. More importantly, security policies can be applied in real time as these events are streamed, and alert and SIEM processing can be immediately applied.

15.1.1 Homegrown audit reporting

As with any homegrown software solution, users are faced with a long-term commitment to maintaining what becomes mission-critical code. This code could require specific technical skills. This knowledge might consist of hard to find skills, which in almost all cases need to be developed and cultivated by the customer. How much in-house talent exists with the skill sets necessary to develop and maintain a homegrown auditing solution? The availability of these resources can affect the timeline and quality of a homegrown solution. What if the person who wrote your application leaves the job? To ensure that adequate expertise is available to support a homegrown solution, cross-training and SME retention can become an issue. To demonstrate a separation of roles and responsibility, source code management and program preparation and implementation needs to be tightly controlled, more so than typical application source code.

In addition, homegrown code, while not necessarily complicated, is not trivial in nature and has the potential for being sensitive to changes in hardware, microcode, operating system, and DBMS. InfoSphere Guardium S-TAP for DB2 for z/OS, as a supported Silicon Valley Labs (SVL) managed OTC product, is subjected to rigorous regression testing as new versions of DB2 and z/OS are released. Because customer developed code is subject to the same potential exposures, close scrutiny of service bulletins, PTF cover letters, and announcement materials is required to achieve advance notification of potential impacts to the homegrown program.

Unlike most *typical* application programs, regression testing requirements becomes more frequent as the customer introduces change to their execution environment through hardware, microcode, and software service upgrades. One other consideration is that poor coding choices might result in homegrown code that while currently executable will become incompatible with newer releases of DB2. Although IBM ensures continued execution of InfoSphere Guardium S-TAP for DB2 for z/OS across these changes, it is a customer's responsibility to adapt their homegrown processes as the DBMS environment evolves.

Customer adoption of a homegrown solution might also be subject to interpretation as to how well it conforms to industry standards such as PCI-DSS. Depending on the auditing implementation, the fact that a homegrown approach is stored in source code format and subject to exposure could also be viewed as problematic.

15.1.2 Database activity monitoring with InfoSphere Guardium for DB2 for z/OS

InfoSphere Guardium helps to satisfy most requirements for monitoring and alerting without impacting SLAs and performance, and without requiring changes to databases or applications. Some of the capabilities for database activity and audit reporting and management provided by Guardium for z/OS include:

- Streamlines compliance processes with workflow automation that automatically distributes compliance reports to oversight teams for electronic sign-offs, escalations, and comments.
- Provides a scalable, multitier architecture that easily grows to handle increased workloads, and additional applications and data center locations.
- Prevents unauthorized changes to database schemas and data.

- Allows for the implementation of automated change control reconciliation by comparing approved change tickets with actual changes implemented by your DBAs.
- Includes a best practices library of hundreds of pre-configured reports and policies for SOX, PCI, and data privacy laws such as HIPAA, as well as an easy to use, drag-and-drop *Builder* for creating custom reports and policies.

15.1.3 InfoSphere Guardium heterogeneous database support

The focus of this book so far has been a discussion of capabilities relating to securing the DB2 for z/OS environment and describing the new security features of DB2 10. However, when looking at compliance and security across the enterprise, few installations are only storing data on a single DBMS platform. Indeed, to satisfy unique requirements for line of business capabilities, as a result of acquisitions or mergers, or the requirement for maintaining earlier applications, most customers have multiple different DBMS platforms deployed.

In almost all cases, when a DBMS is used to store sensitive data, the regulatory requirements and the supporting auditing requirements from a security and audit standpoint should be consistent and, as much as possible, identical. However, the reality, as auditing is implemented across disparate DBMS platforms, seems to be directly opposite from this ideal. Typically, each DBMS type is configured by separate groups, with separate views of how to best comply with security and audit reporting requirements, while having to work within the capabilities and restrictions of the native DBMS security and audit logging mechanisms.

The root cause for this situation is pretty clear, and the result is that each DBMS platform views access and transaction processing, from a security and audit standpoint, in different ways. DBMS platform differences result in different metrics and audit event information being collected, and different reporting mechanisms then being used to generate activity and audit reports.

However, because the security and audit department in most organizations take a consolidated view of security across the *entire* enterprise, there are yet additional steps that need to be taken to combine audit and security event information into a single consumable point to realistically use this data for every day monitoring and protection of databases. But, for many customers, a comprehensive auditing environment requires much more than just collecting, storing, and providing reporting mechanisms. Most customers today require support for heterogeneous DBMS environments, spread across different hardware, operating systems, and database managers. Effective auditing implies that data from these disparate environments be combined in a *single pane* view.

InfoSphere Guardium for z/OS, when combined in a larger heterogeneous implementation, provides additional support that satisfies many of the common auditing and reporting requirements, as well as providing significantly more robust functionality.

Figure 15-1 shows the heterogeneous database support provided by InfoSphere Guardium.

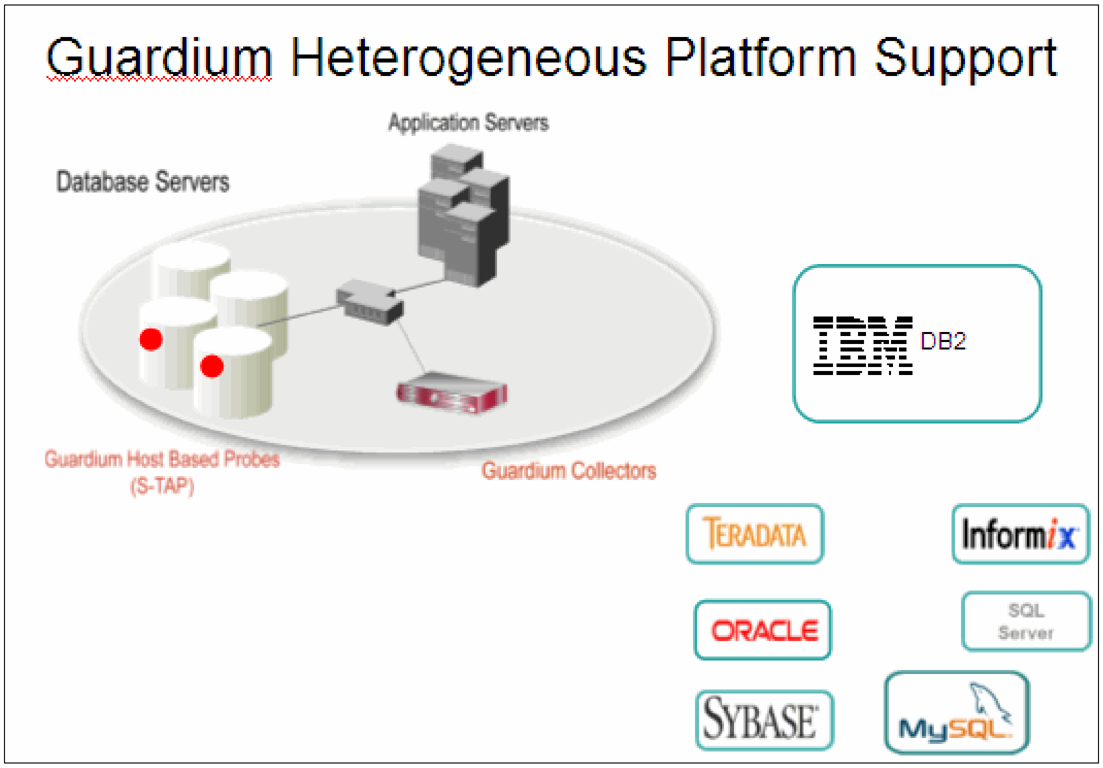


Figure 15-1 InfoSphere Guardium heterogeneous database support

InfoSphere Guardium provides the capability to support a wide range of DBMS platforms running on different operating systems. Auditors using InfoSphere Guardium for z/OS do not need to go to a large number of sources to access data. They simply log in to InfoSphere Guardium for z/OS to gain complete visibility of all auditable objects. An auditor can display collected data for all DB2 instances, or just the DB2 instances of interest, all from the central repository.

A central repository creates a single source for reporting, institutional controls, summarization of the data, including high-level trending of audit anomalies and the ability to drill down (one layer at a time), and a robust level of reporting events controlled by the auditor without DBA involvement. You secure audit data in a *locked down*, tamper-proof audit repository that cannot be modified by anyone, including DBAs and other privileged users, thereby supporting separation of duties and addressing a key requirement from auditors.

When audit data resides in a hardened environment like InfoSphere Guardium for z/OS, customers can further control access and better protect their audit data.

15.1.4 InfoSphere Guardium components in a heterogeneous environment

InfoSphere Guardium, when deployed in an enterprise solution, consists of a number of different components, some optional, and their presence is dependent on the specific auditing and database monitoring requirements needed.

Figure 15-2 shows the various InfoSphere Guardium components deployed in a typical environment.

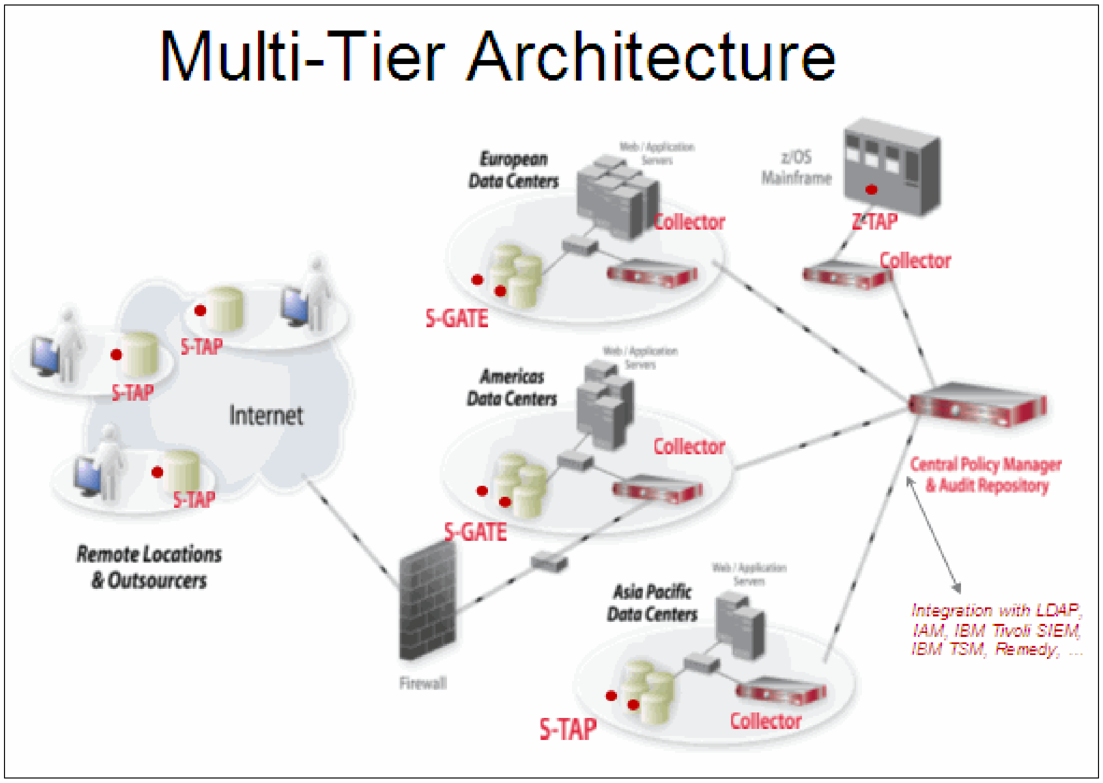

Multi-Tier Architecture
European Data Centers
Web / Application Servers
z/OS Mainframe
Collector
Z-TAP
Collector
S-GATE
S-TAP
S-TAP
Internet
Americas Data Centers
Web / Application Servers
Collector
Central Policy Manager & Audit Repository
S-TAP
S-GATE
Remote Locations & Outsourcers
Asia Pacific Data Centers
Web / Application Servers
Integration with LDAP, IAM, IBM Tivoli SIEM, IBM TSM, Remedy, ...
Firewall
S-TAP
Collector

Figure 15-2 Multitiered architecture

There are several different hardware components called “collectors”. The InfoSphere Guardium server (Central Policy Manager) is a high-performance Linux-based appliance developed by Guardium for efficient off-mainframe analysis and storage of massive amounts of audit data. The appliance provides sufficient self-contained capacity for online storage of up to 4 billion database transactions with built-in purging, archiving, and aggregating capabilities to effectively manage the data collected. InfoSphere Guardium appliances are shipped with the software installed, and with an initial configuration specified during the purchase process. The Linux-based appliance is shipped with a “hardened” operating system, which means no root access is provided, and with no native database access or privileges to the audit repository stored on the appliance.

InfoSphere Guardium S-TAP is a lightweight software agent installed on a database server system. It monitors database traffic and forwards information about that traffic to a InfoSphere Guardium appliance, which can be deployed anywhere in the network. Because it is installed on a database server system, InfoSphere Guardium S-TAP can monitor database traffic that is local to that system. This function is important because local connections can provide “back door” access to the database and all such access needs to be monitored and audited. In addition to monitoring local connections, InfoSphere Guardium S-TAP can be used to monitor any network traffic that is visible from the database server on which it is installed. It can thus act as a collector on remote network segments, where it is not practical to install a InfoSphere Guardium appliance.

InfoSphere Guardium S-TAP collects and sends data to a InfoSphere Guardium host in near real time. InfoSphere Guardium S-TAP buffers the data, so that it can continue to work if the InfoSphere Guardium host is momentarily unavailable. If the primary host is unavailable for an extended period of time, InfoSphere Guardium S-TAP can fail over to a secondary InfoSphere Guardium host. It continues to send data to the secondary host until either that appliance becomes unavailable, or until the InfoSphere Guardium S-TAP is restarted, at which point it attempts to connect to its primary InfoSphere Guardium host first.

InfoSphere Guardium S-TAP, when configured as a firewall, has the unique capability that allows it to screen connections; this configuration is referred to as S-GATE. A connection can be watched, in which case rules defined on the InfoSphere Guardium server are used to evaluate each interaction with the database. Through the S-GATE Terminate action, a new external security overlay can be added to control access, regardless of which database platform is used. Moreover, this mechanism can be used for all connection types (remote *and* local) and for all users, limiting even administrators and users with SYSADM or SYSTEM DBADM privileges from accessing sensitive data.

Aggregation allows you to collect and merge information from multiple InfoSphere Guardium servers to a single InfoSphere Guardium aggregation server. If you are running InfoSphere Guardium in an enterprise deployment, you may have multiple InfoSphere Guardium servers monitoring different environments (different geographic locations or business units, for example). It may be useful to collect all the data in a central location to facilitate an enterprise view of database usage. You can accomplish this task by exporting data from a number of servers to another server that has been configured (during the initial installation procedures) as an aggregation server. In such a deployment, you typically run all reports, assessments, audit processes, and so on, on the aggregation server to achieve an enterprise view.

InfoSphere Guardium also supports hierarchical aggregation, where multiple aggregation units merge upwards to a higher-level, central aggregation server. This function is useful for multi-level views. For example, you may need to deploy one aggregation server for North America to aggregate multiple units, another aggregation server for Asia to aggregate multiple units, and a central, global aggregation server merging the contents of the North America and Asia aggregation servers into a single corporate view. To consolidate data, all aggregated InfoSphere Guardium servers export data to the aggregation server on a scheduled basis. The aggregation server imports that data into a single database on the aggregation server, so that reports run on the aggregation server are based on the data consolidated from all of the aggregated InfoSphere Guardium servers.

In a central management configuration, one InfoSphere Guardium unit is designated as the Central Manager. That unit can be used to monitor and control other InfoSphere Guardium units, which are referred to as managed units. Un-managed units are referred to as stand-alone units. The concept of a *local machine* can refer to any machine in the Central Management system. There are some applications (Audit Processes, Queries, Portlets, and so on) that can be run on both the Managed Units and the Central Manager. In both cases, the definitions come from the Central Manager and the data comes from the local machine (which could also be the Central Manager). After a Central Management system is set up, customers can use either the Central Manager or a Managed Unit to create or modify most definitions. Keep in mind that most of the definitions reside on the Central Manager, regardless of on which machine the actual auditing is done.

Users access the InfoSphere Guardium appliance over a secure (HTTPS) connection using a web browser. All users are defined to the system by the InfoSphere Guardium access manager. Your access manager provides you with your user name and initial password. In a default installation, you need to change your password the first time that you log in.

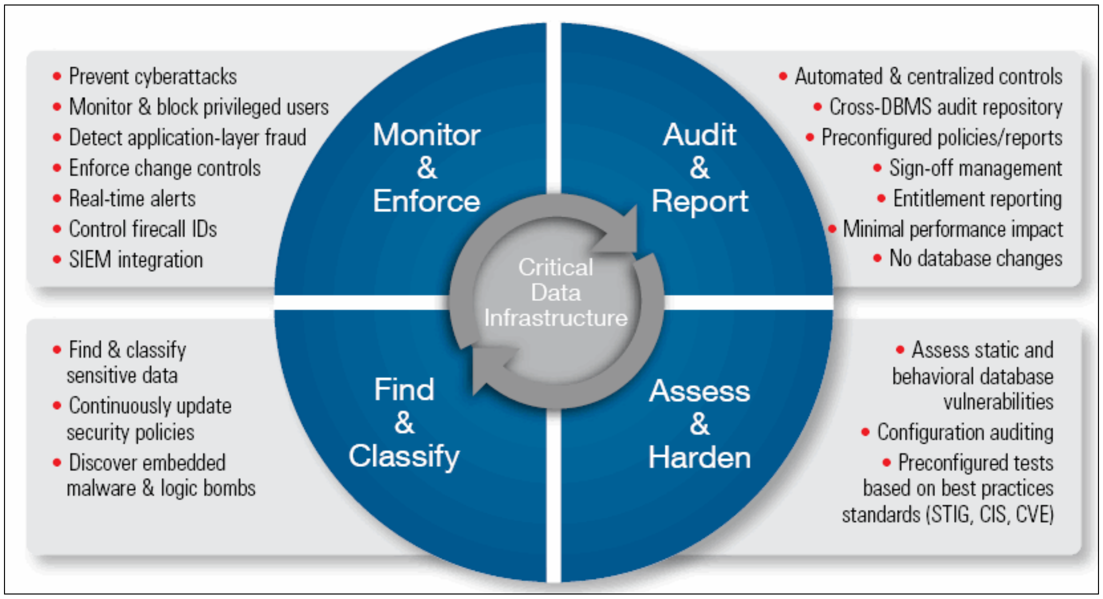
Within your browser window, InfoSphere Guardium displays information in a set of tabbed panes. By selecting a tab or menu option, you can overlay panes on the same screen area. Each pane may contain one or more portlets. A portlet can be a report, application, or tool. Each pane may contain any number of report portlets, and a single application or tool portlet. When you log in for the first time, your portal displays with a layout determined by the InfoSphere Guardium roles that the access manager has assigned to your user account. Although the access manager controls the initial layout, you can customize your layout easily, changing the panes displayed and the placement of portlets on each pane.

The management of users and roles is usually reserved for the access manager, that is, the InfoSphere Guardium user who is assigned the accessmgr user name. Defining and modifying users involves deciding both who will be using the InfoSphere Guardium system and to what roles they will be assigned. When getting started with a InfoSphere Guardium appliance, an important early task is to identify which groups of users will use that appliance, and what their function will be. For example, an information security group might use InfoSphere Guardium for alerting and troubleshooting purposes; a database administrator group might use InfoSphere Guardium for reporting and monitoring. When deciding who will access the InfoSphere Guardium system, keep in mind that sensitive company data can be picked up by the system. Therefore, be aware of who will be able to access that data.

15.2 Database security functionality using InfoSphere Guardium

As described earlier in this chapter, using the native trace facility of DB2 10 for z/OS provides a source of audit trail events. It is up to the security function in each individual organization to describe what constitutes meaningful reporting criteria, create the audit reporting components, and then, once implemented, manage the ongoing collection and reporting process.

With the InfoSphere Guardium framework, the collection of events are now policy driven, and does not rely on the use of native DBMS trace facilities. In addition, with the capabilities shown in Figure 15-3, you can implement a life cycle based security framework.



• Prevent cyberattacks
• Monitor & block privileged users
• Detect application-layer fraud
• Enforce change controls
• Real-time alerts
• Control firecall IDs
• SIEM integration
Monitor & Enforce
Audit & Report
• Automated & centralized controls
• Cross-DBMS audit repository
• Preconfigured policies/reports
• Sign-off management
• Entitlement reporting
• Minimal performance impact
• No database changes
Critical Data Infrastructure
• Find & classify sensitive data
• Continuously update security policies
• Discover embedded malware & logic bombs
Find & Classify
Assess & Harden
• Assess static and behavioral database vulnerabilities
• Configuration auditing
• Preconfigured tests based on best practices standards (STIG, CIS, CVE)

Figure 15-3 InfoSphere Guardium and the security life cycle

Some of the capabilities provided by the InfoSphere Guardium solution include:

- Vulnerability Assessment provides a set predefined and custom tests, along with a process workflow, allowing organizations to identify and address database vulnerabilities in an automated fashion, proactively improving configurations and hardening infrastructures. Database Vulnerability Assessment scans the database infrastructure for vulnerabilities and provides evaluation of database and data security health, with real-time and historical measurements.
- Databases can be affected by changes to the server environment, for example, by changing configuration files, environment or registry variables, or other database or operating system components, including executables or scripts used by the database management system or the operating system. The Change Audit System (CAS) tracks such changes and reports them. The data is available on the InfoSphere Guardium appliance and can be used for reports and alerts.
- InfoSphere Guardium provides the Classification feature to discover and classify sensitive data, so that you can make and enforce effective access policy decisions. As the size and organization of the corporate database grows, sensitive information such as credit card numbers and transactions, or personal financial data, might be present in multiple locations, without the knowledge of the current owners of that data. This frequently happens in corporations that have experienced mergers and acquisitions and in older corporations where earlier systems have outlasted their original owners. Even in the best of cases, integration and enhancement projects between disparate systems can easily leave sensitive data unknown and unprotected. Classification support in InfoSphere Guardium provides the ability to establish classification policies, which are a set of rules designed to discover and tag sensitive data elements (database tables or files). A set of actions can be specified to be taken when each rule is triggered. An action might be to generate an email alert, or to add a member to a group. Each time a rule is satisfied, that event is logged, and thus can be reported upon.

- Many customers have valuable information in many different databases in their environment. It is extremely useful for a audit report to correlate relevant information to make these reports easy and useful to understand. The External Data Connector allows users to create custom tables on the InfoSphere Guardium appliance for enterprise information that is needed in addition to the existing InfoSphere Guardium internal data. This action can be done either manually within the GUI or based on an existing table on a database server. Queries and reports can then be created for this information just as though it were predefined data.
- The InfoSphere Guardium Value Change Auditing feature tracks changes to values in database tables. For each table in which changes are to be tracked, you select which SQL value-change commands to monitor (insert, update, or delete). Each time a value-change command is executed against a monitored table, before and after values are captured. On a scheduled basis, the change activity is uploaded to a InfoSphere Guardium appliance, where all of the InfoSphere Guardium reporting and alerting functions can be used.
- InfoSphere Guardium Report Builder and Query Editor help organize the data collected by the InfoSphere Guardium server into a set of domains. Each domain contains a different type of information relating to a specific area of concern: data access, exceptions, policy violations, and so on. A report defines how the data collected by a query is presented. The default report is a tabular report that reflects the structure of the query, with each attribute displayed in a separate column. All presentation components of a tabular report (the column headings, for example) can be customized using the Report Builder. In addition, all graphical reports are defined using the Report Builder.

The InfoSphere Guardium cross-platform solution supports all major DBMS platforms and protocols running on all major operating systems (Windows, UNIX, Linux, and z/OS), as well as Microsoft SharePoint and FTP environments. For details about its capabilities by product and platform, go to the following address:

http://www.ibm.com/software/data/guardium/

15.3 InfoSphere Guardium S-TAP for DB2 for z/OS

InfoSphere Guardium S-TAP for DB2 for z/OS (also referred to as InfoSphere and S-TAP) collects and correlates data access information from a variety of DB2 sources to produce a comprehensive view of business activity for auditors.

InfoSphere Guardium operates in a client-server environment using a combination of one or more servers, agents, administration repositories, and data collectors. InfoSphere Guardium S-TAP for DB2 for z/OS is composed of several components that work together to collect audit DB2 activity for DB2 subsystems. Figure 15-4 shows these components.

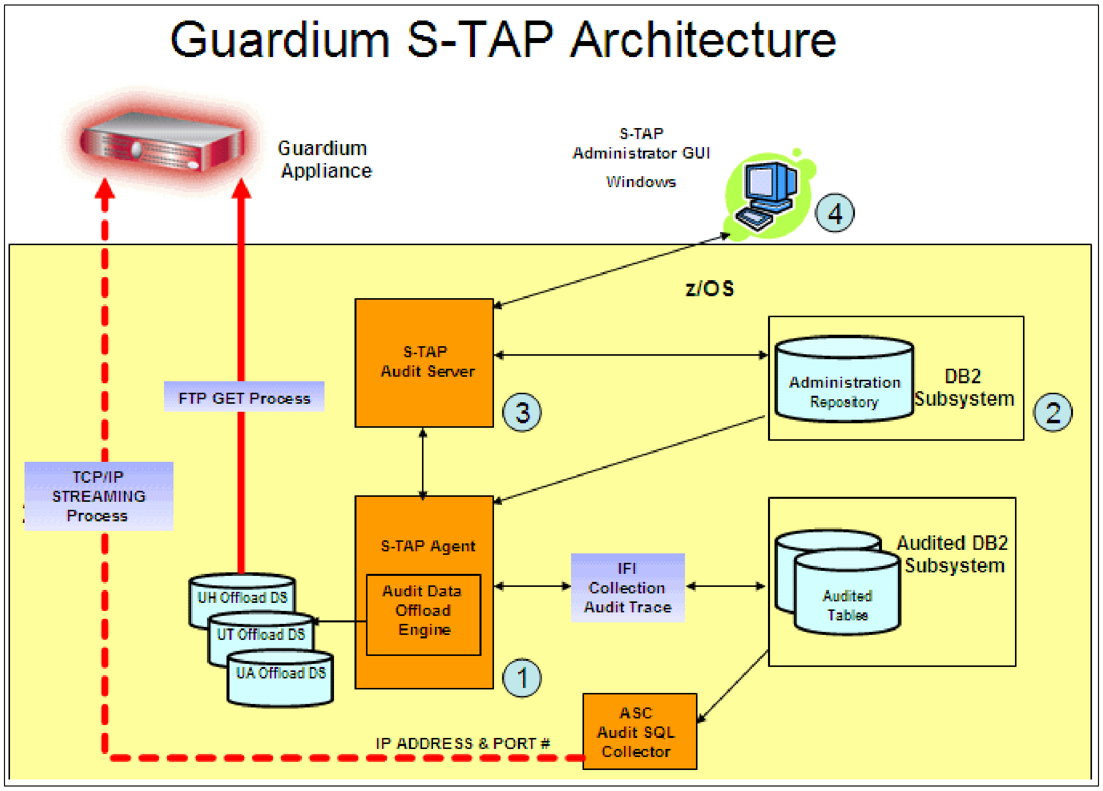


Figure 15-4 InfoSphere Guardium S-TAP architecture

Where:

1. The InfoSphere agent is responsible for the collection of audit data in an InfoSphere environment. The InfoSphere agent runs as a started task on z/OS and acts as a "container" to run the various collectors that are appropriate to the specific type of system on which it operates. The InfoSphere agent captures audit trace information, and writes to offload files (for IFI-collected data) or streams it to a InfoSphere Guardium appliance (for ASC-collected data). The agent also maintains the necessary communications link to send information back and forth to the InfoSphere server address space.
2. The administration repository is DB2-based and contains tables to define DB2 systems and their properties. Each administration repository is associated with an InfoSphere server and is standalone (meaning repositories do not interact with other InfoSphere servers or repositories). Multiple administration repositories within the InfoSphere environment are permitted. The administration repository *only* stores profile filtering and S-TAP runtime execution data; it is not used to store audit event information.
3. The audit server is the central control point for an InfoSphere network. A single audit server, running as a started task on z/OS, can support data collection from multiple agents on multiple z/OS systems, including multiple DB2 subsystems. The audit server provides services to:
   - Support the administration user interface.
   - Access the administration repository.

  – Set up monitoring criteria.
  – Perform repository administration functions

4. InfoSphere provides a GUI-based administration user interface. The administration user interface enables InfoSphere product administrators to perform administrative tasks such as creating collections and collection profiles, and adding and modifying administration users.

To describe the configuration for Infosphere Guardium S-TAP for DB2 for z/OS, if we take a typical environment where we have a single DB2 subsystem in the z/OS LPAR that has an auditing scope, the following components are required:

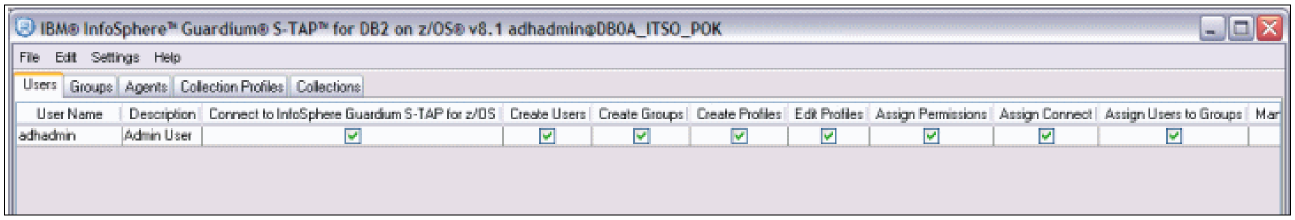
- A single Infosphere Guardium S-TAP for DB2 for z/OS Server address space.
- A single MASTER address space, which is only used as a control block anchor point, so it consumes no CPU and has no system resources allocated.
- A single AGENT address space, which is responsible the collection and management of the IFI based information and starts and stops the ASC address space.
- A single ASC (Audit SQL Collector) address space. The ASC is responsible for “hook” placement and SQL inspection within the DB2 address space.

If we were to introduce a second monitored DB2 subsystem into the previously described z/OS LPAR, we would need these additional components:

- A second AGENT address space
- A second ASC address space

Collection profiles and policy administration

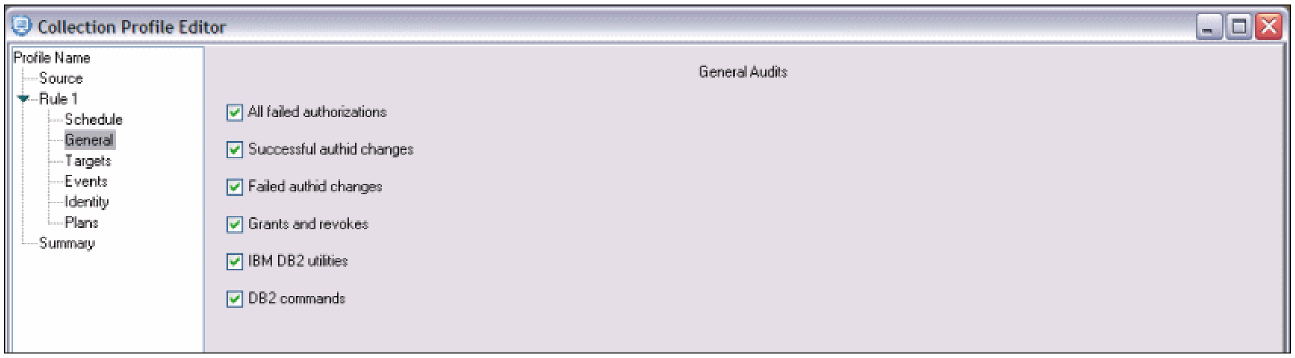
As shown in Figure 15-4 on page 357, the security administrator defines collection profiles using the administration GUI. Administration rights and privileges are group and member based, and are controlled by the administration server. Access to the administration GUI is password controlled, and its use can be additionally restricted by limiting the distribution of the administration client to a small group of authorized users. Figure 15-5 shows the users administration window.


IBM® InfoSphere™ Guardium® S-TAP™ for DB2 on z/OS® v8.1 adhadmin@DB0A_ITSO_POK
File Edit Settings Help
Users Groups Agents Collection Profiles Collections

User Name	Description	Connect to InfoSphere Guardium S-TAP for z/OS	Create Users	Create Groups	Create Profiles	Edit Profiles	Assign Permissions	Assign Connect	Assign Users to Groups	Mar
adhadmin	Admin User	✓	✓	✓	✓	✓	✓	✓	✓	

Figure 15-5 Infosphere Guardium S-TAP for z/OS user administration

Collection profiles determine which events on DB2 for z/OS are of interest and would be within scope for reporting and processing. Infosphere Guardium S-TAP for DB2 for z/OS collects events using two different mechanisms. For all non-SQL related events, the Infosphere Guardium S-TAP Agent address space starts and manages IFI based trace events. These are event types that are characterized as low frequency and low impact events. Figure 15-6 shows the specification of these events.



✓ All failed authorizations
✓ Successful authid changes
✓ Failed authid changes
✓ Grants and revokes
✓ IBM DB2 utilities
✓ DB2 commands

Figure 15-6 General audit event specification in the collection profile

When general audit events are specified, Infosphere Guardium S-TAP for DB2 for z/OS starts audit class traces to the OPx buffer. Table 15-1 associates the trace class with the general audit classification category.

Table 15-1 Audit category and trace class

General audit event category	DB2 trace class
All failed authorizations	Class 1
Successful authid changes	Class 7
Failed authid changed	Class 7
Grants and revokes	Class 2
IBM Utilities	Class 8
DB2 Command	Not collected with IFI trace

For SQL related events, the Audit SQL Collector (ASC) address space inspects each SQL statement as it is processed by DB2, and based on collection profile filters, gathers additional information and externalizes the event, through TCP/IP streaming, to the Infosphere Guardium appliance.

Filters can be applied to a list of authids, objects, or plans. Filters can specify that specific groups of elements be either included or excluded from collection. Figure 15-7 shows a representative object filter.

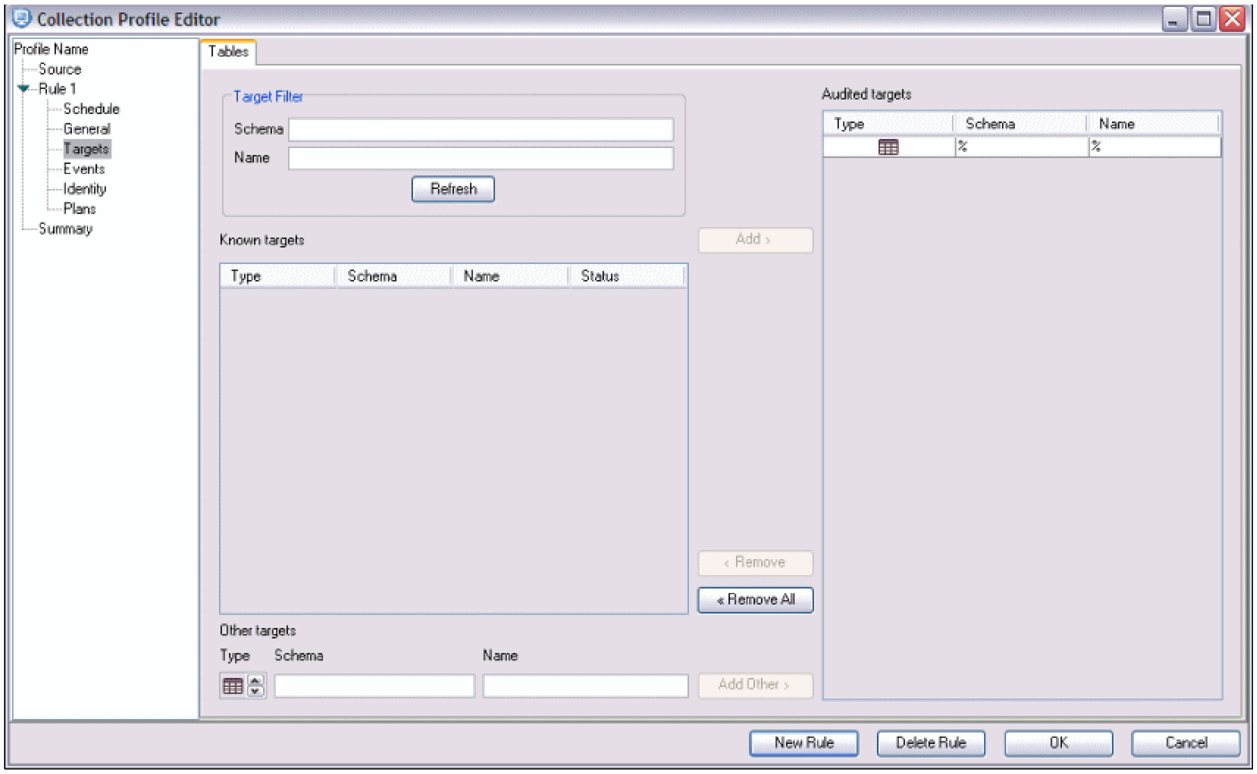
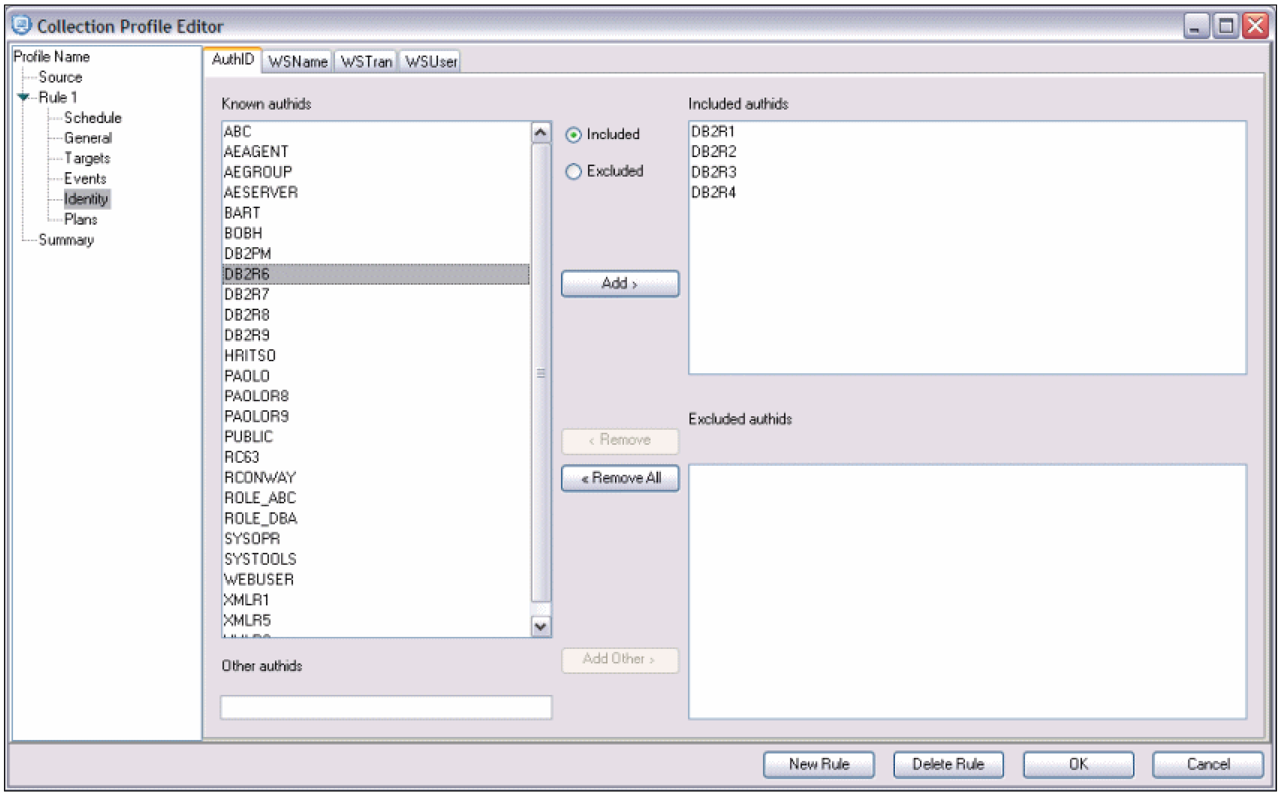


Figure 15-7 *Object target filter specification*

In the above example, both the schema and name use the % wildcard character, which means that any object access is eligible for collection (subject to additional filters). Authorization identifiers can also be the subject of filter specification, as shown in Figure 15-8.



Known authids: ABC, AEAGENT, AEGROUP, AESERVER, BART, BOBH, DB2PM, DB2R6, DB2R7, DB2R8, DB2R9, HRITSO, PAOLO, PAOLOR8, PAOLOR9, PUBLIC, RC63, RCONWAY, ROLE_ABC, ROLE_DBA, SYSOPR, SYSTOOLS, WEBUSER, XMLR1, XMLR5

Included authids: DB2R1, DB2R2, DB2R3, DB2R4

Figure 15-8 Authorization identifier filters

The purpose of the ID-Only filtering is to produce CPU reduction for collection and filtering for certain auditing situations. ID-Only filtering can be used in situations involving the auditing of a subset of users (for example, SYSADMs). ID-Only filtering allows the mechanism in InfoSphere to decide as early as possible in the filtering process whether to audit a specific SQL event.

The filtering of SQL events against rules in an active collection profile occurs in two stages:

- Stage 1 filtering occurs at the point of collection for a limited set of identifying fields.
- Stage 2 filtering occurs during ‘post collection' processing of the event.

For Stage 1-only filtering to occur, all of the rules in the active collection profile must contain only IDs and only when targets are specified with (*Schema = ‘%' and Name = ‘%'*) and Read or Changes events is checked. All SQL events are filtered in Stage 1 at the point of collection, which means that no unaudited events are passed to Stage 2 processing.

For the customer who is primarily focused on the monitoring of privileged users, ID-Only filtering can provide an appreciable CPU reduction.

In addition, for the customer who has zIIP specialty processors deployed in their environment, some of the filtering workload is zIIP offload eligible, and can be offloaded when a zIIP is available. This offload capability is user specified and controlled through a specification in the ASC configuration member.

15.3.1 Event collection

After the collection profile has been saved, and the policy has been activated, SQL event collection and publication can be performed. As mentioned previously, there are two types of collection and publication. The first type of publication, which applies to the general audit event classes collected using the DB2 IFI trace, is performed by using the TCP/IP FTP protocol. Figure 15-9 shows the general process.

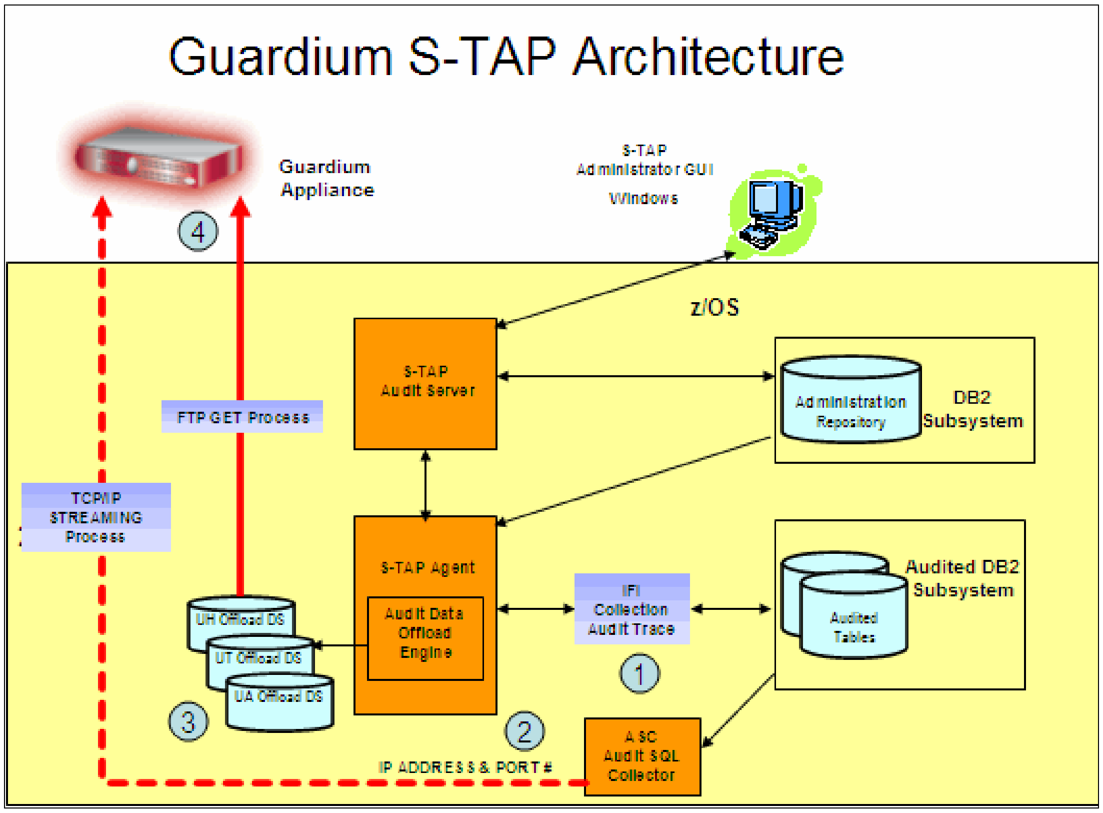
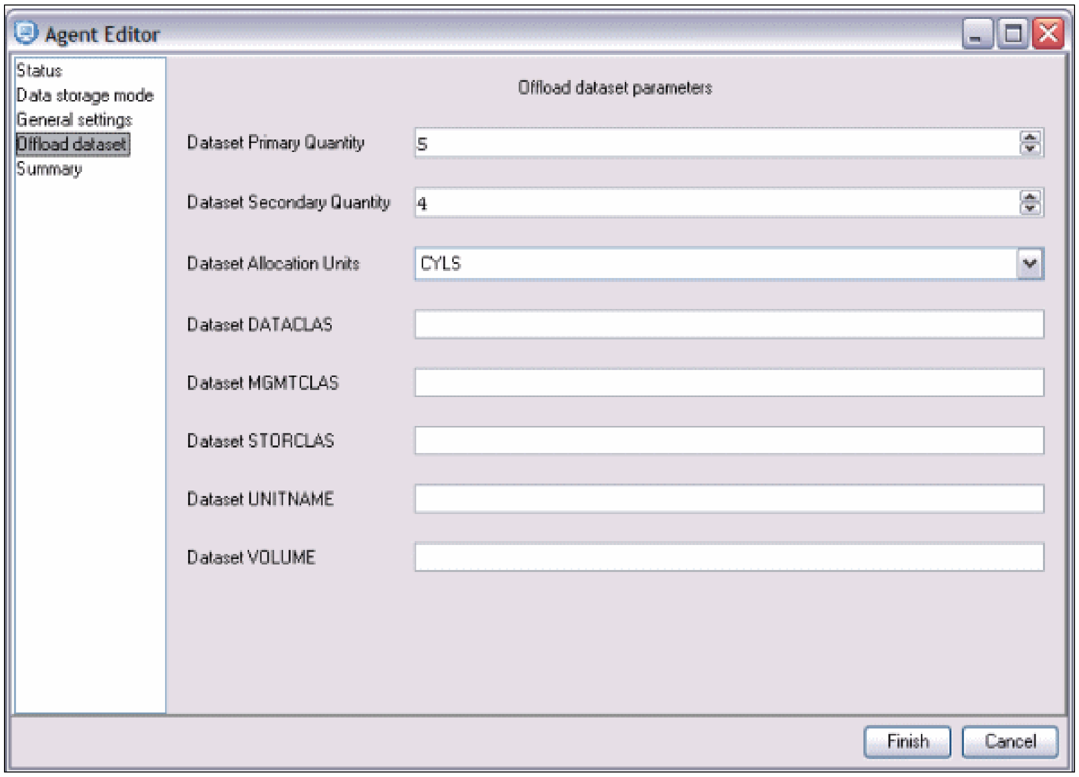


Figure 15-9 InfoSphere Guardium FTP process

Where:

1. The FTP process starts by collecting the IFI trace event and sending it to the OPx buffer destination.
2. The event is formatted and written to a staging data set. At an interval, typically 30 minutes, but configured within the Infosphere Guardium Agent policy, these staging data sets are renamed, and a new set of staging data sets are allocated. The renamed data sets are now referred to as offload data sets, and are eligible for processing.
3. At an interval defined on the Infosphere Guardium appliance, an FTP process is initiated to retrieve the offload data sets, using either SFTP or FTP.
4. On the Infosphere Guardium appliance, the offload data events are processed, and ultimately event information is inserted into the various Infosphere Guardium audit tables located on the appliance. After the offload data set has been successfully processed, the appliance then deletes the offload data set from the z/OS host server.

Data set name, allocation characteristics, and other data set parameters are controlled by the Infosphere Guardium agent policy definition. Figure 15-10 shows a representative example.



Offload dataset parameters

Dataset Primary Quantity	5
Dataset Secondary Quantity	4
Dataset Allocation Units	CYLS
Dataset DATACLAS	
Dataset MGMTCLAS	
Dataset STORCLAS	
Dataset UNITNAME	
Dataset VOLUME	

Figure 15-10 Offload data set characteristics in a policy definition

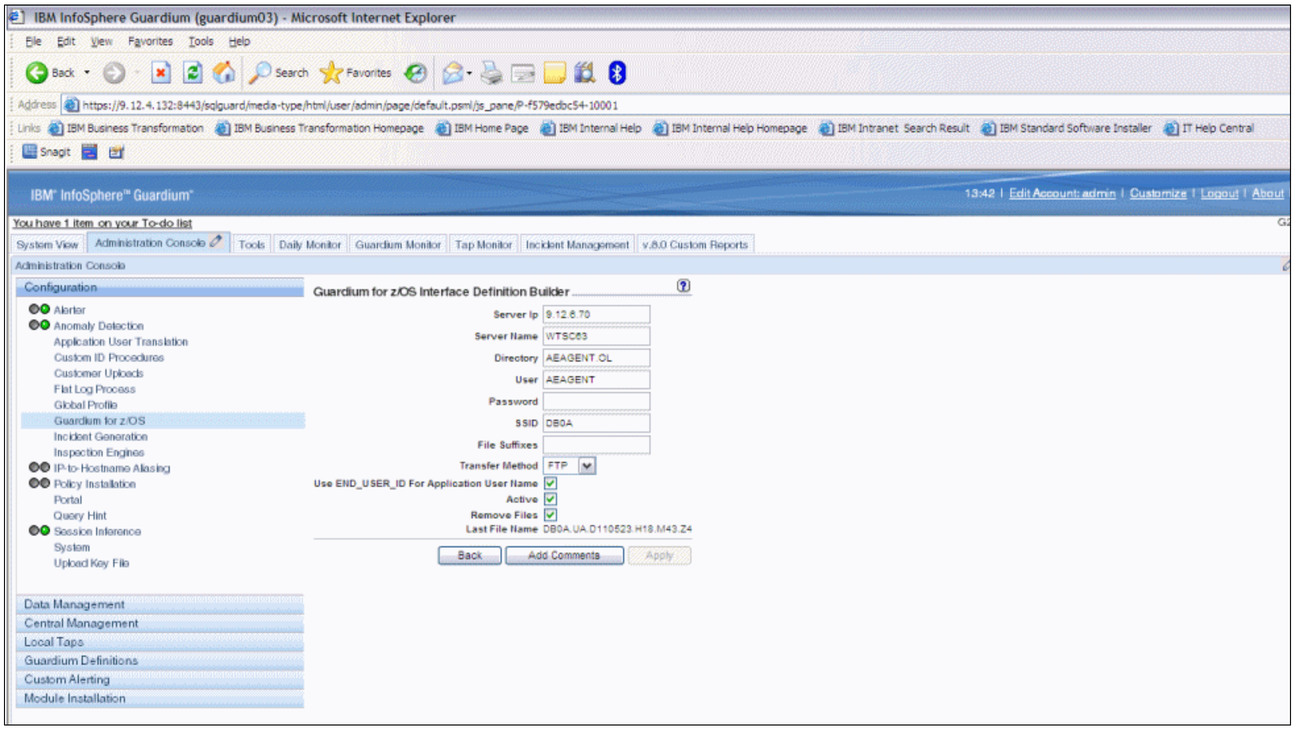
Whenever a collection interval is reached, controlled by the Infosphere Guardium S-TAP agent policy, the offload data sets are created as defined in the offload data set characteristics. They take the form of the data set names shown in Example 15-1.

Example 15-1 Representative offload data set name

```
AEAGENT.OL.DB0A.UA.D110524.H13.M34.T5
AEAGENT.OL.DB0A.UT.D110524.H13.M34.T5
```

After events are collected and externalized into the offload data set, the data is then consumed by the Infosphere Guardium appliance through FTP. The FTP configuration is performed by the Infosphere Guardium administrator, who interacts with the appliance using a web-based administration GUI. Users access the Infosphere Guardium appliance over a secure (HTTPS) connection, and all users are defined to the system by the Infosphere Guardium access manager. The use of this GUI, and control over who has administration privileges on the appliance, is determined by your access manager.

In the Infosphere Guardium for z/OS Interface Definition Builder (Figure 15-11), the following parameters are used to define how the FTP process is initiated from the appliance to the z/OS server.



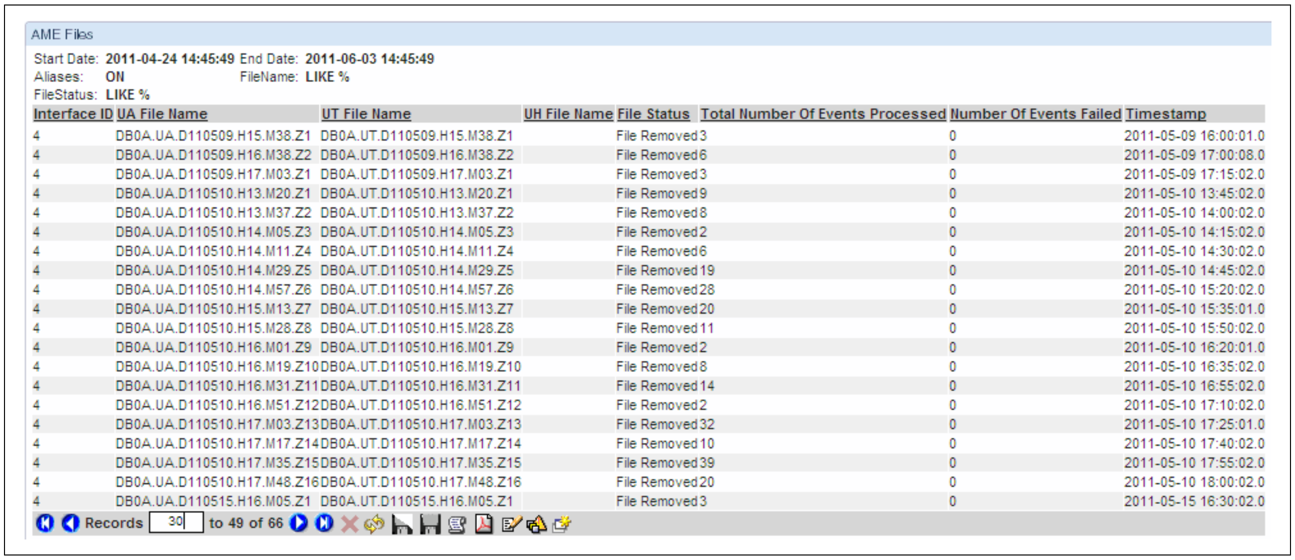
Guardium for z/OS Interface Definition Builder

Server Ip	9.12.6.70
Server Name	WTSC63
Directory	AEAGENT.OL
User	AEAGENT
Password	
SSID	DB0A
File Suffixes	
Transfer Method	FTP
Use END_USER_ID For Application User Name	✓
Active	✓
Remove Files	✓
Last File Name	DB0A.UA.D110523.H18.M43.Z4

Figure 15-11   FTP configuration from the appliance

- *Server IP* is the IP address of the z/OS host.
- *Server Name* is the host name of the z/OS host.
- *Directory* is the first two nodes of the offload data set name shown in Example 15-1 on page 363.
- *User* is a RACF ID with FTP authority to the z/OS host, as well as the ability to read and delete the offload data sets.
- *Password* is the password associated with the RACF ID previously described.
- *SSID* is the DB2 subsystem name from where the events were collected.
- *Remove Files* specifies that the offload data sets are to be deleted when processed.

Once configured, the FTP process occurs on a default schedule of 15 minutes. This frequency can be controlled by the Infosphere Guardium appliance administrator. As offload data sets are successfully consumed, there is a process log kept on the Infosphere Guardium appliance that shows the history of offload data set processing, which is shown in Figure 15-12.


AME Files

Start Date: **2011-04-24 14:45:49** End Date: **2011-06-03 14:45:49**
Aliases: **ON** FileName: **LIKE %**
FileStatus: **LIKE %**

Interface ID	UA File Name	UT File Name	UH File Name	File Status	Total Number Of Events Processed	Number Of Events Failed	Timestamp
4	DB0A.UA.D110509.H15.M38.Z1	DB0A.UT.D110509.H15.M38.Z1		File Removed	3	0	2011-05-09 16:00:01.0
4	DB0A.UA.D110509.H16.M38.Z2	DB0A.UT.D110509.H16.M38.Z2		File Removed	6	0	2011-05-09 17:00:08.0
4	DB0A.UA.D110509.H17.M03.Z1	DB0A.UT.D110509.H17.M03.Z1		File Removed	3	0	2011-05-09 17:15:02.0
4	DB0A.UA.D110510.H13.M20.Z1	DB0A.UT.D110510.H13.M20.Z1		File Removed	9	0	2011-05-10 13:45:02.0
4	DB0A.UA.D110510.H13.M37.Z2	DB0A.UT.D110510.H13.M37.Z2		File Removed	8	0	2011-05-10 14:00:02.0
4	DB0A.UA.D110510.H14.M05.Z3	DB0A.UT.D110510.H14.M05.Z3		File Removed	2	0	2011-05-10 14:15:02.0
4	DB0A.UA.D110510.H14.M11.Z4	DB0A.UT.D110510.H14.M11.Z4		File Removed	6	0	2011-05-10 14:30:02.0
4	DB0A.UA.D110510.H14.M29.Z5	DB0A.UT.D110510.H14.M29.Z5		File Removed	19	0	2011-05-10 14:45:02.0
4	DB0A.UA.D110510.H14.M57.Z6	DB0A.UT.D110510.H14.M57.Z6		File Removed	28	0	2011-05-10 15:20:02.0
4	DB0A.UA.D110510.H15.M13.Z7	DB0A.UT.D110510.H15.M13.Z7		File Removed	20	0	2011-05-10 15:35:01.0
4	DB0A.UA.D110510.H15.M28.Z8	DB0A.UT.D110510.H15.M28.Z8		File Removed	11	0	2011-05-10 15:50:02.0
4	DB0A.UA.D110510.H16.M01.Z9	DB0A.UT.D110510.H16.M01.Z9		File Removed	2	0	2011-05-10 16:20:01.0
4	DB0A.UA.D110510.H16.M19.Z10	DB0A.UT.D110510.H16.M19.Z10		File Removed	8	0	2011-05-10 16:35:02.0
4	DB0A.UA.D110510.H16.M31.Z11	DB0A.UT.D110510.H16.M31.Z11		File Removed	14	0	2011-05-10 16:55:02.0
4	DB0A.UA.D110510.H16.M51.Z12	DB0A.UT.D110510.H16.M51.Z12		File Removed	2	0	2011-05-10 17:10:02.0
4	DB0A.UA.D110510.H17.M03.Z13	DB0A.UT.D110510.H17.M03.Z13		File Removed	32	0	2011-05-10 17:25:01.0
4	DB0A.UA.D110510.H17.M17.Z14	DB0A.UT.D110510.H17.M17.Z14		File Removed	10	0	2011-05-10 17:40:02.0
4	DB0A.UA.D110510.H17.M35.Z15	DB0A.UT.D110510.H17.M35.Z15		File Removed	39	0	2011-05-10 17:55:02.0
4	DB0A.UA.D110510.H17.M48.Z16	DB0A.UT.D110510.H17.M48.Z16		File Removed	20	0	2011-05-10 18:00:02.0
4	DB0A.UA.D110515.H16.M05.Z1	DB0A.UT.D110515.H16.M05.Z1		File Removed	3	0	2011-05-15 16:30:02.0

Figure 15-12 Infosphere Guardium S-TAP processed files log

The second category of events collected by Infosphere Guardium S-TAP for DB2 for z/OS are those events related to SQL activity. The collection of these events are performed by the Audit SQL Collector (ASC) component. The ASC is a started task that is controlled by the InfoSphere Guardium S-TAP for DB2 for z/OS agent.

At startup, the ASC component installs intercepts within DB2 that perform an inspection of each SQL statement processed by DB2. Based on the installed collection profile, if a particular SQL statement is determined to be within the audit scope, information describing this statement is collected from the relevant DB2 control blocks, and an audit event record is formatted. After the event record has been constructed, it is then streamed to the InfoSphere Guardium appliance by the ASC address space through TCP/IP streaming.

Latency: General audit events are published through FTP, with some user-defined latency between collection and publication to the InfoSphere Guardium appliance. The SQL events collected by the ASC components are immediately streamed through TCP/IP with little or no latency. Events are immediately available for alert and reporting.

The InfoSphere Guardium appliance is configured within a customer network with an IP address, and a listener port assignment, which by default is 16016. The InfoSphere Guardium S-TAP for DB2 for z/OS TCP/IP process is configured through specifications in the ASC parameter member, to connect to this IP address.

Example 15-2 shows the representative configuration values.

Example 15-2 Connection by ASC to InfoSphere Guardium appliance

```
APPLIANCE_SERVER(9.12.4.132) -
APPLIANCE_PORT(16016)   -
APPLIANCE_PING_RATE(15)      -
APPLIANCE_RETRY_INTERVAL(15) -
APPLIANCE_CONNECT_RETRY_COUNT(10) -
APPLIANCE_NETWORK_REQUEST_TIMEOUT(500) -
```

Once connected, events are consumed and processed by the appliance as soon as they are published. The ASC started task can display the number of SQL events inspected and the number of events that have been shipped to the appliance, as shown in Example 15-3.

Example 15-3 Event collection statistics from the S-TAP command on ASC

```
ADHQ3270I STAP INFO: STAGE1 FILTER IS.............. ACTIVE
ADHQ3270I STAP INFO: STAGE2 FILTER IS.............. ACTIVE
ADHQ3270I STAP INFO: TOTAL SQL CALLS DETECTED....... 0000000000000052
ADHQ3270I STAP INFO: PASSED STAGE1 FILTERING........ 000000000000000D
ADHQ3270I STAP INFO: AUDS BLOCKS QUEUED............ 000000000000000A
ADHQ3270I STAP INFO: AUDS BLOCKS PASSED STAGE2...... 000000000000000A
ADHQ3270I STAP INFO: AUDS BLOCKS SENT TO APPLIANCE.. 000000000000000A
ADHQ3270I STAP INFO: AUDS BLOCKS NOT SENT TO APPL... 0000000000000000
ADHQ3270I STAP INFO: AUDS BLOCKS FREED ............ 000000000000000A
ADHQ3270I STAP INFO: AUDS BLOCKS FREED LOST ........ 0000000000000000
ADHQ3270I STAP INFO: BYTES SENT.................... 000000000001E08C
```

15.3.2 SQL inspection and performance monitoring

As previously described, SQL related events are collected by the ASC component through control block level inspection processing. In order for any audit and activity reporting solution to be considered robust, from a compliance perspective, all SQL must be inspected. However, in a production environment, with a significant velocity of SQL activity, this inspection impact, in some cases, is not negligible.

For many SQL performance monitors, there is a similar SQL inspection process that is used to collect SQL performance metrics. When collecting audit related information and performance instrumentation, the application impact can, in many cases, be the sum of both inspection events.

To reduce the overall impact to the monitored environment, when using InfoSphere Guardium S-TAP for DB2 for z/OS in conjunction with Query Monitor V2.3 or greater, both products share the same SQL inspection component (ASC). In a configuration where the requirement is for the collection of both audit events and performance information, this combination of solutions can provide a significant decrease in CPU consumption for the SQL inspection process.

For more information about Query Monitor, refer to *DB2 Query Monitor for z/OS User Guide,* SC18-9202.

15.3.3 Building audit reports with InfoSphere Guardium

A domain provides a view of the stored data in the InfoSphere Guardium audit repository. Each domain contains a set of data related to a specific purpose or function (data access, exceptions, policy violations, and so on). Each domain contains one or more entities. An entity is a set of related attributes, and an attribute is basically a field value.

A query returns data from one domain only. When the query is defined, one entity within that domain is designated as the main entity of the query. Each row of data returned by a query contains a count of occurrences of the main entity matching the values returned for the selected attributes, for the requested time period. This allows for the creation of two-dimensional reports from entities that do not have a one-to-one relationship.

A query describes a set of information to be obtained from the collected data, for example, “find all clients updating a specific database during weekend hours.” A report describes how the data returned by a query is presented. Most often, the report is in tabular form, but extensive graphical reporting capabilities are provided as well.

Figure 15-13 shows the entity list associated with the flat log domain.

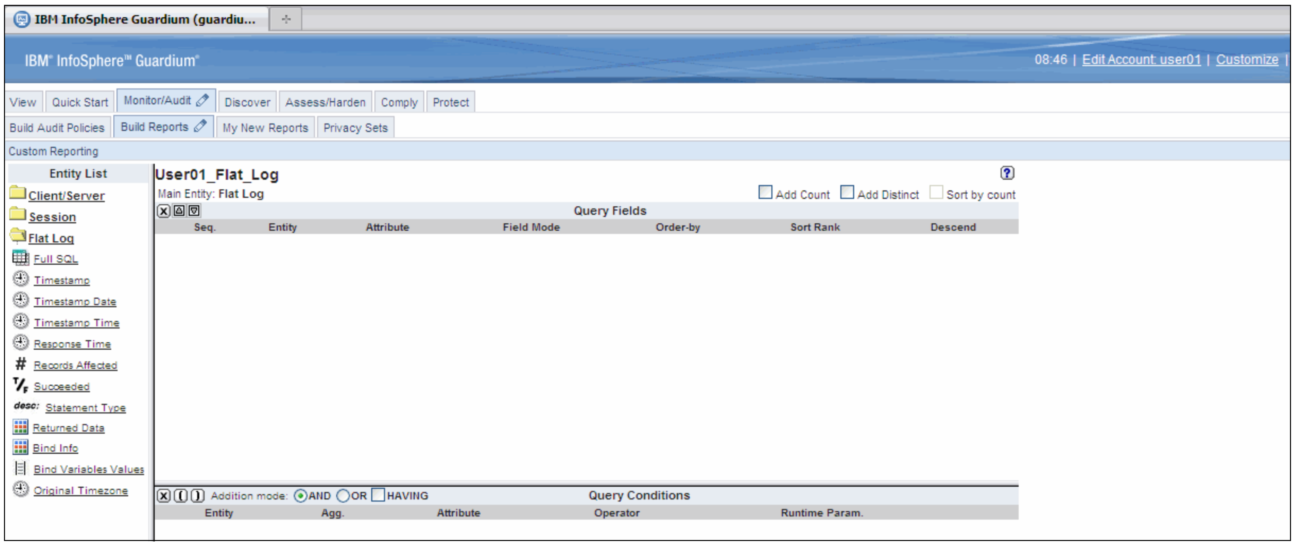


Figure 15-13 Query builder - Entity List

The Flat Log option is a process to allow the Infosphere Guardium appliance to log information without immediately parsing it in real time. This process saves resources, so that a heavier traffic volume can be handled. The entity list folders on the left show the different elements that can be selected to create the query for the report being built. Elements are selected by clicking the element, then choosing to add the element as either a field or a condition, as shown in Figure 15-14.

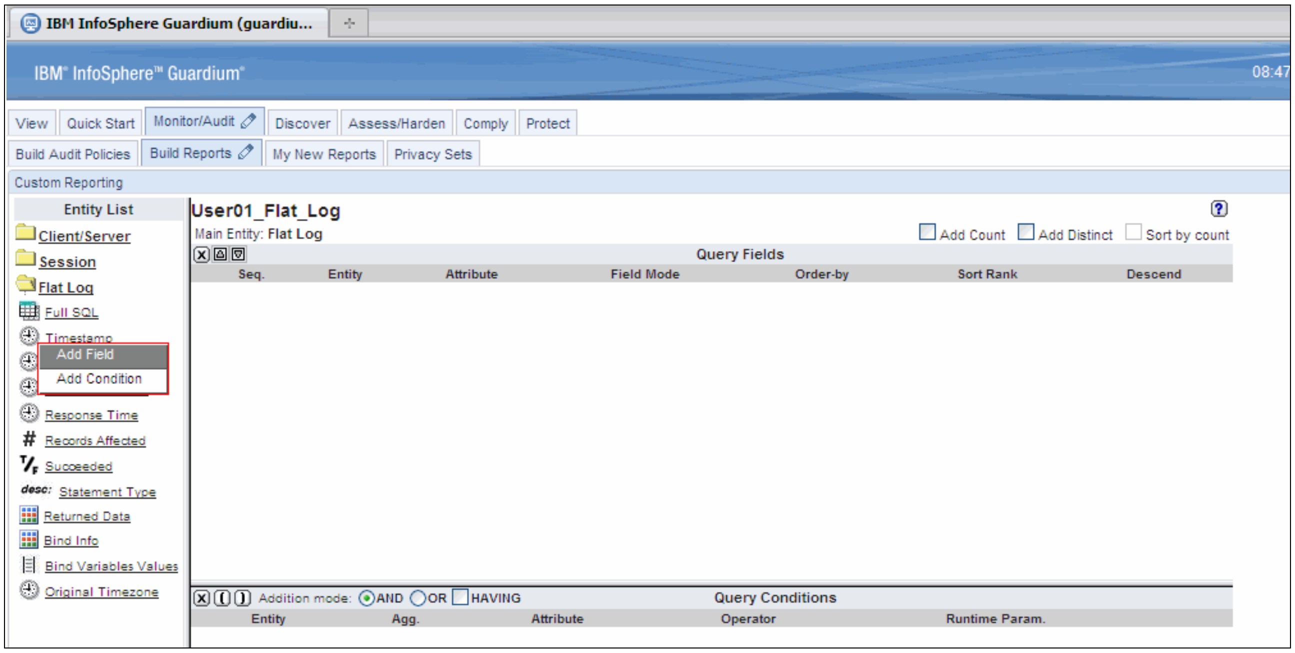
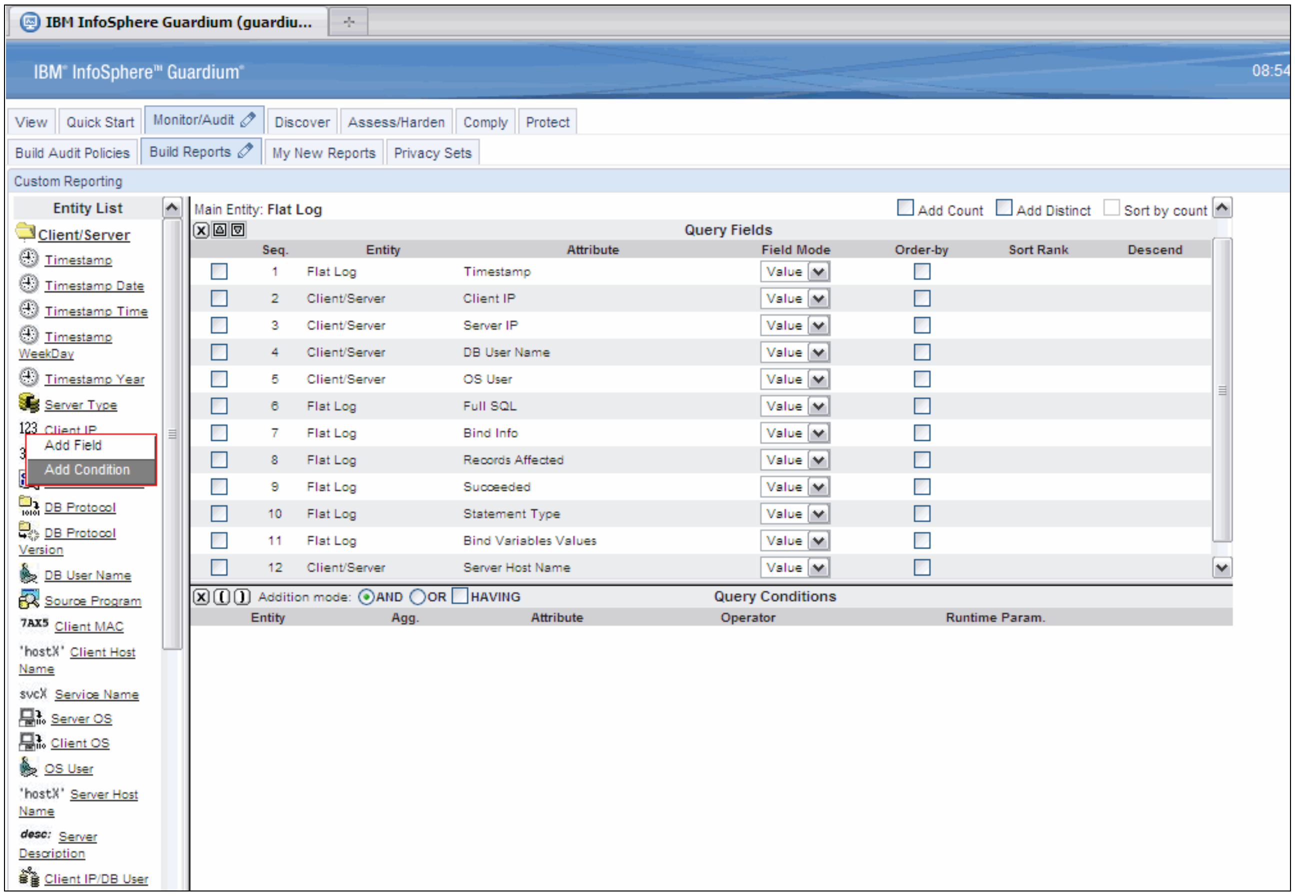


Figure 15-14 Adding elements as fields to a query

You may add as many fields as desired to generate your query. Some elements are shown as individual values, while others are present as counts. Figure 15-15 shows the query with all the desired elements selected.



Seq.	Entity	Attribute	Field Mode	Order-by	Sort Rank	Descend
1	Flat Log	Timestamp	Value	☐		
2	Client/Server	Client IP	Value	☐		
3	Client/Server	Server IP	Value	☐		
4	Client/Server	DB User Name	Value	☐		
5	Client/Server	OS User	Value	☐		
6	Flat Log	Full SQL	Value	☐		
7	Flat Log	Bind Info	Value	☐		
8	Flat Log	Records Affected	Value	☐		
9	Flat Log	Succeeded	Value	☐		
10	Flat Log	Statement Type	Value	☐		
11	Flat Log	Bind Variables Values	Value	☐		
12	Client/Server	Server Host Name	Value	☐		

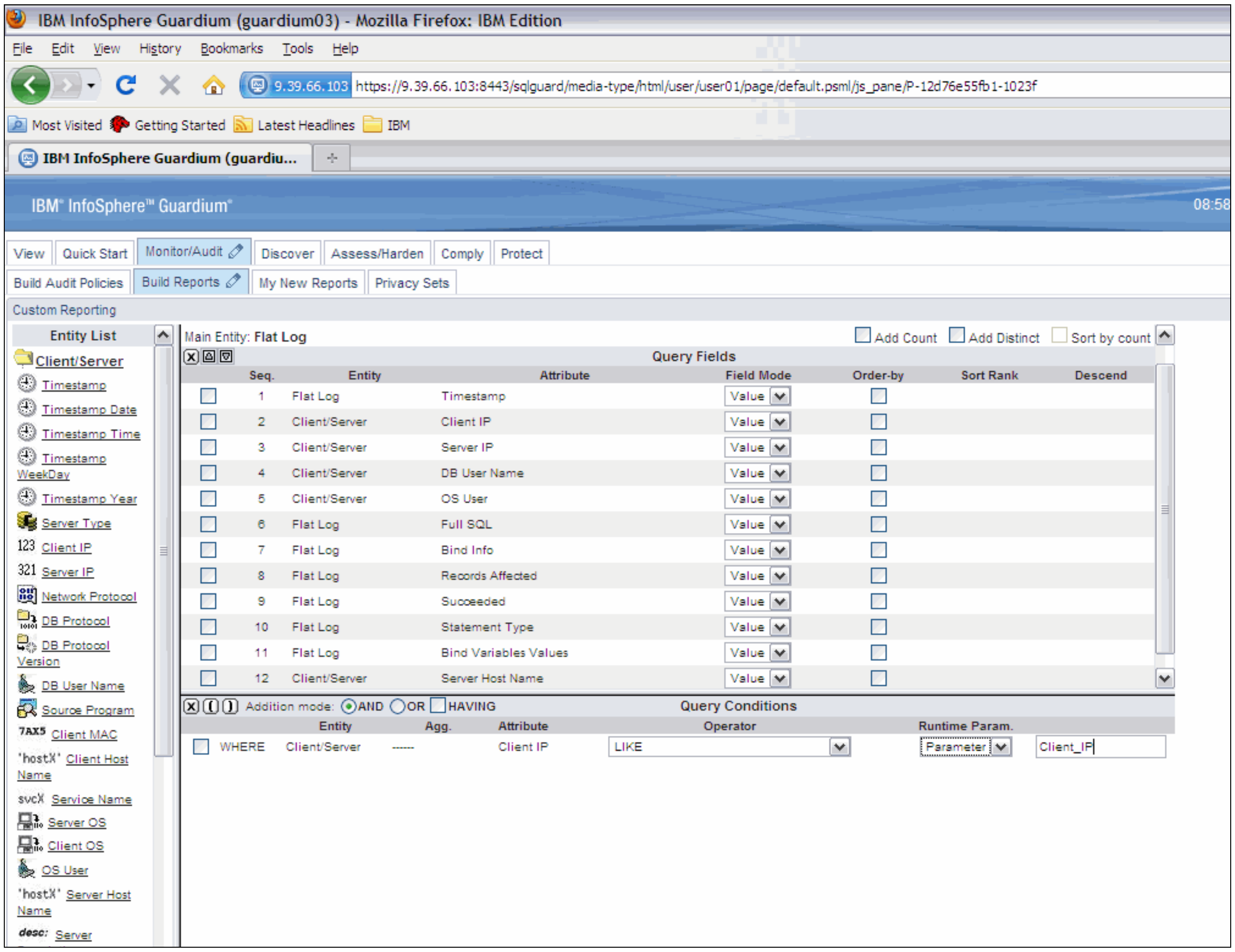
Entity	Agg.	Attribute	Operator	Runtime Param.

Figure 15-15 Query with multiple elements selected

Some elements can be shown as values, some as min/max, and this mode is selected in the query form. In addition, elements can be specified as part of an order-by clause, with multiple elements used, and ascending or descending order specified.

After the query elements have been selected, you can then select elements to be used as part of the query qualification. These are known as *query conditions*. Query conditions are used to control what data is selected from the data domain.

To select an element as a condition, click the element in the entity list folder, and select the **Add Condition** button. The window shown in Figure 15-16 opens.


IBM InfoSphere Guardium (guardium03) - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
https://9.39.66.103:8443/sqlguard/media-type/html/user/user01/page/default.psml/js_pane/P-12d76e55fb1-1023f
Most Visited Getting Started Latest Headlines IBM
IBM InfoSphere Guardium (guardiu...
IBM InfoSphere Guardium
08:58
View Quick Start Monitor/Audit Discover Assess/Harden Comply Protect
Build Audit Policies Build Reports My New Reports Privacy Sets
Custom Reporting
Entity List
Client/Server
Timestamp
Timestamp Date
Timestamp Time
Timestamp WeekDay
Timestamp Year
Server Type
Client IP
Server IP
Network Protocol
DB Protocol
DB Protocol Version
DB User Name
Source Program
Client MAC
Client Host Name
Service Name
Server OS
Client OS
OS User
Server Host Name
Server
Add Count
Add Distinct
Sort by count

Main Entity: Flat Log

Query Fields

	Seq.	Entity	Attribute	Field Mode	Order-by	Sort Rank	Descend
☐	1	Flat Log	Timestamp	Value	☐		
☐	2	Client/Server	Client IP	Value	☐		
☐	3	Client/Server	Server IP	Value	☐		
☐	4	Client/Server	DB User Name	Value	☐		
☐	5	Client/Server	OS User	Value	☐		
☐	6	Flat Log	Full SQL	Value	☐		
☐	7	Flat Log	Bind Info	Value	☐		
☐	8	Flat Log	Records Affected	Value	☐		
☐	9	Flat Log	Succeeded	Value	☐		
☐	10	Flat Log	Statement Type	Value	☐		
☐	11	Flat Log	Bind Variables Values	Value	☐		
☐	12	Client/Server	Server Host Name	Value	☐		

Addition mode: [✓ AND] OR ☐ HAVING

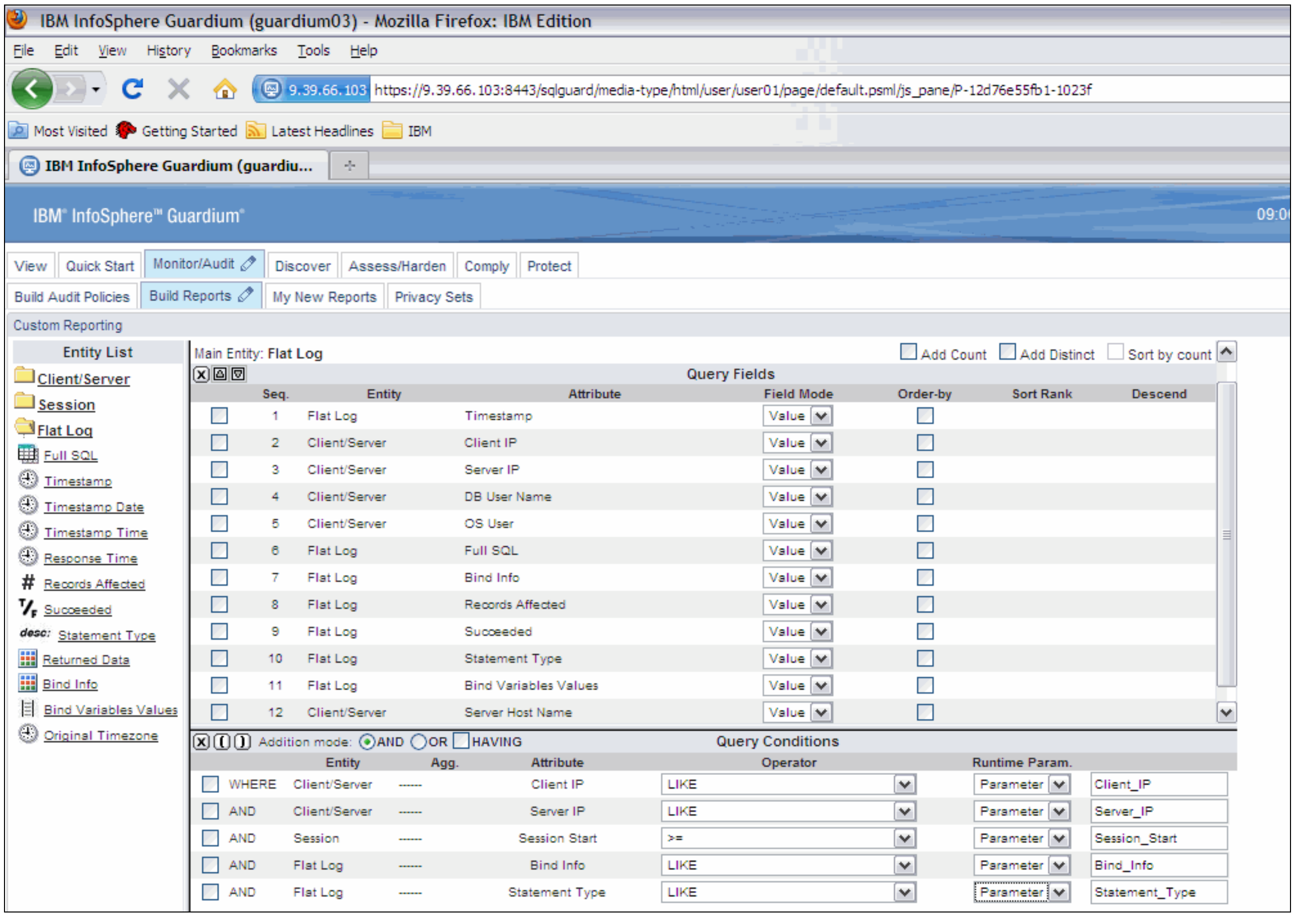
Query Conditions

		Entity	Agg.	Attribute	Operator	Runtime Param.	
☐	WHERE	Client/Server	------	Client IP	LIKE	Parameter	Client_IP

Figure 15-16 Adding conditions to the query

Conditions can be acted on using various operators while following the SQL convention. In addition, the condition can be a hardcoded literal, or can be specified as a parameter, which can be changed at run time as reporting requirements dictate. In our example, we specify our conditions as runtime parameters.

Figure 15-17 shows the completed query with 5 different runtime parameters specified.


IBM InfoSphere Guardium (guardium03) - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
9.39.66.103 https://9.39.66.103:8443/sqlguard/media-type/html/user/user01/page/default.psml/js_pane/P-12d76e55fb1-1023f
Most Visited Getting Started Latest Headlines IBM
IBM InfoSphere Guardium (guardiu...
IBM InfoSphere™ Guardium
View Quick Start Monitor/Audit Discover Assess/Harden Comply Protect
Build Audit Policies Build Reports My New Reports Privacy Sets
Custom Reporting
Entity List
Client/Server
Session
Flat Log
Full SQL
Timestamp
Timestamp Date
Timestamp Time
Response Time
Records Affected
Succeeded
Statement Type
Returned Data
Bind Info
Bind Variables Values
Original Timezone
Add Count Add Distinct Sort by count

Main Entity: Flat Log

Query Fields

Seq.	Entity	Attribute	Field Mode	Order-by	Sort Rank	Descend
1	Flat Log	Timestamp	Value			
2	Client/Server	Client IP	Value			
3	Client/Server	Server IP	Value			
4	Client/Server	DB User Name	Value			
5	Client/Server	OS User	Value			
6	Flat Log	Full SQL	Value			
7	Flat Log	Bind Info	Value			
8	Flat Log	Records Affected	Value			
9	Flat Log	Succeeded	Value			
10	Flat Log	Statement Type	Value			
11	Flat Log	Bind Variables Values	Value			
12	Client/Server	Server Host Name	Value			

Addition mode: [✓ AND] OR HAVING

Query Conditions

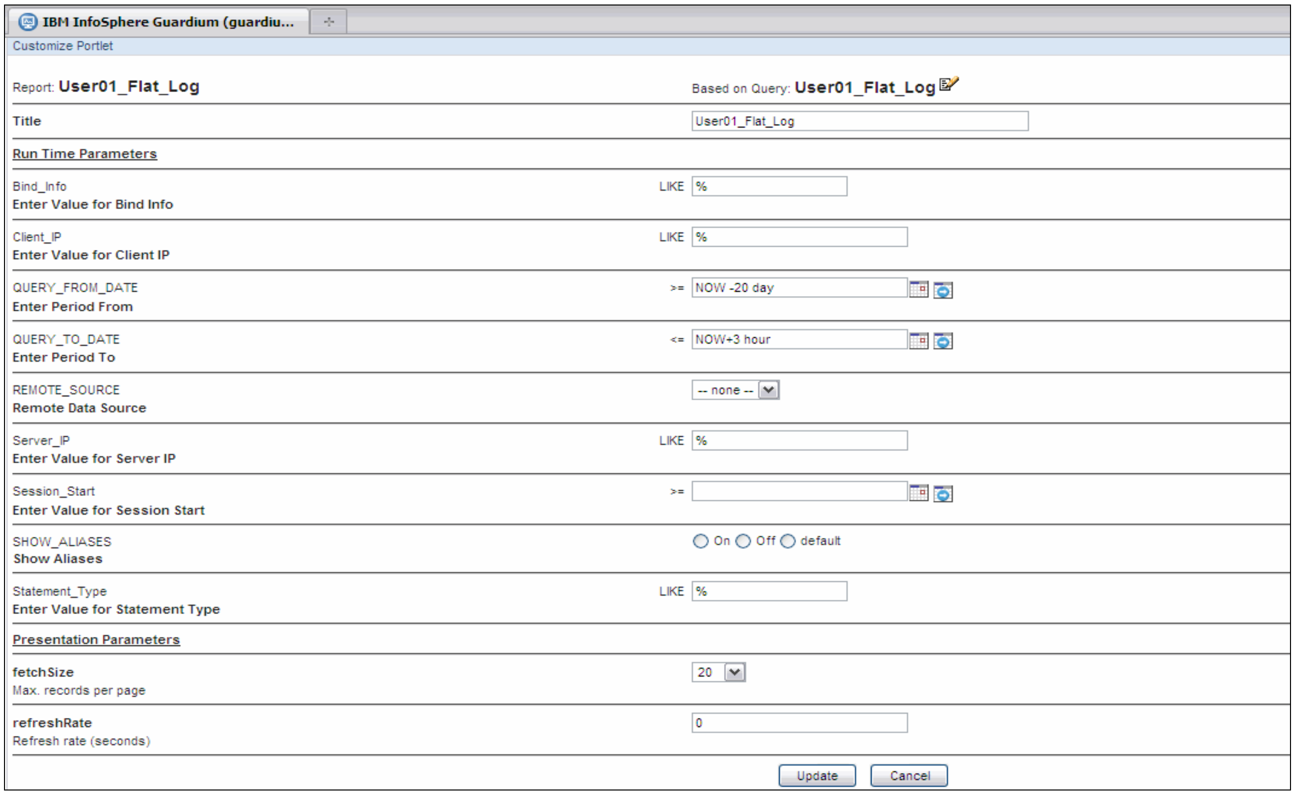
	Entity	Agg.	Attribute	Operator	Runtime Param.	
WHERE	Client/Server	------	Client IP	LIKE	Parameter	Client_IP
AND	Client/Server	------	Server IP	LIKE	Parameter	Server_IP
AND	Session	------	Session Start	>=	Parameter	Session_Start
AND	Flat Log	------	Bind Info	LIKE	Parameter	Bind_Info
AND	Flat Log	------	Statement Type	LIKE	Parameter	Statement_Type

Figure 15-17 Completed query with runtime conditions

After the query has been saved, the report writer is used to generate the audit report, and place the report on one or more tabs. Reports can be designated as public, and accessible to all users of Infosphere Guardium, or can be limited to use by authorized users or groups.

Prior to execution, the runtime parameters need to be designated. These values can represent values, or you can use wildcards using the standard SQL escape character.

Figure 15-18 show the specification of runtime values for the parameters in this query.



IBM InfoSphere Guardium (guardiu...

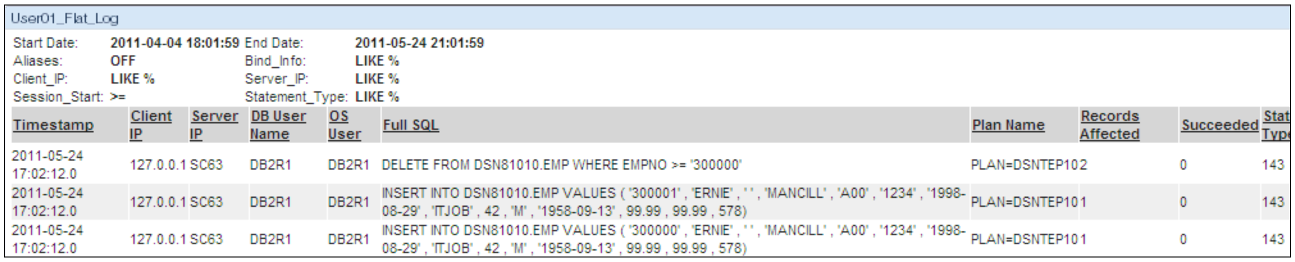
Customize Portlet

Report: **User01_Flat_Log**	Based on Query: **User01_Flat_Log**
Title	User01_Flat_Log
Run Time Parameters	
Bind_Info Enter Value for Bind Info	LIKE %
Client_IP Enter Value for Client IP	LIKE %
QUERY_FROM_DATE Enter Period From	>= NOW -20 day
QUERY_TO_DATE Enter Period To	<= NOW+3 hour
REMOTE_SOURCE Remote Data Source	-- none --
Server_IP Enter Value for Server IP	LIKE %
Session_Start Enter Value for Session Start	>=
SHOW_ALIASES Show Aliases	○ On ○ Off ○ default
Statement_Type Enter Value for Statement Type	LIKE %
Presentation Parameters	
fetchSize Max. records per page	20
refreshRate Refresh rate (seconds)	0

Update Cancel

Figure 15-18 Specification of runtime values for query execution

As an example, we repeated the dynamic SQL example described in Example 12-5 on page 279. When running the InfoSphere Guardium report, we see the report output shown in Figure 15-19.



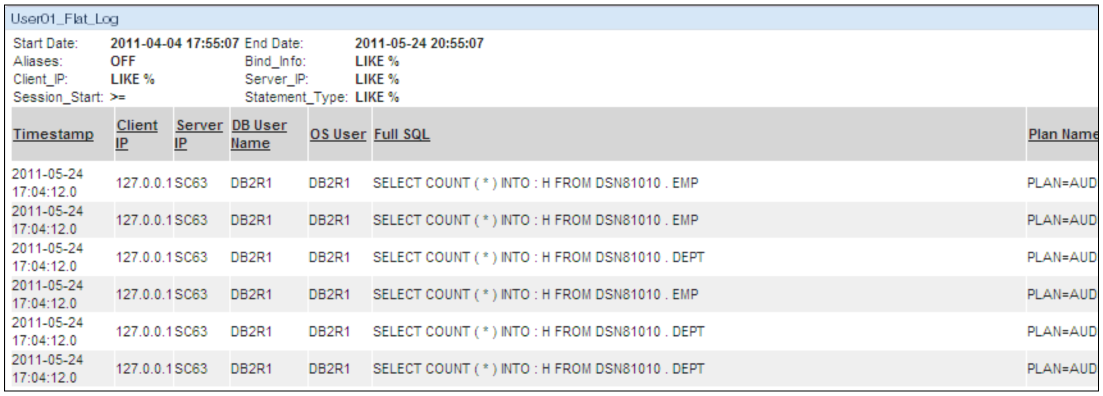
User01_Flat_Log

Start Date:	**2011-04-04 18:01:59**	End Date:	**2011-05-24 21:01:59**
Aliases:	**OFF**	Bind_Info:	**LIKE %**
Client_IP:	**LIKE %**	Server_IP:	**LIKE %**
Session_Start:	**>=**	Statement_Type:	**LIKE %**

Timestamp	Client IP	Server IP	DB User Name	OS User	Full SQL	Plan Name	Records Affected	Succeeded	Stat Typ
2011-05-24 17:02:12.0	127.0.0.1	SC63	DB2R1	DB2R1	DELETE FROM DSN81010.EMP WHERE EMPNO >= '300000'	PLAN=DSNTEP10	2	0	143
2011-05-24 17:02:12.0	127.0.0.1	SC63	DB2R1	DB2R1	INSERT INTO DSN81010.EMP VALUES ('300001' , 'ERNIE' , ' ' , 'MANCILL' , 'A00' , '1234' , '1998-08-29' , 'ITJOB' , 42 , 'M' , '1958-09-13' , 99.99 , 99.99 , 578)	PLAN=DSNTEP10	1	0	143
2011-05-24 17:02:12.0	127.0.0.1	SC63	DB2R1	DB2R1	INSERT INTO DSN81010.EMP VALUES ('300000' , 'ERNIE' , ' ' , 'MANCILL' , 'A00' , '1234' , '1998-08-29' , 'ITJOB' , 42 , 'M' , '1958-09-13' , 99.99 , 99.99 , 578)	PLAN=DSNTEP10	1	0	143

Figure 15-19 Dynamic SQL scenario in Infosphere Guardium report

We also ran the static workload using the COBOL program described in Example 12-4 on page 278. The InfoSphere Guardium report section is shown in Figure 15-20.


User01_Flat_Log

Start Date:	2011-04-04 17:55:07	End Date:	2011-05-24 20:55:07
Aliases:	OFF	Bind_Info:	LIKE %
Client_IP:	LIKE %	Server_IP:	LIKE %
Session_Start:	>=	Statement_Type:	LIKE %

Timestamp	Client IP	Server IP	DB User Name	OS User	Full SQL	Plan Name
2011-05-24 17:04:12.0	127.0.0.1	SC63	DB2R1	DB2R1	SELECT COUNT (*) INTO : H FROM DSN81010 . EMP	PLAN=AUD
2011-05-24 17:04:12.0	127.0.0.1	SC63	DB2R1	DB2R1	SELECT COUNT (*) INTO : H FROM DSN81010 . EMP	PLAN=AUD
2011-05-24 17:04:12.0	127.0.0.1	SC63	DB2R1	DB2R1	SELECT COUNT (*) INTO : H FROM DSN81010 . DEPT	PLAN=AUD
2011-05-24 17:04:12.0	127.0.0.1	SC63	DB2R1	DB2R1	SELECT COUNT (*) INTO : H FROM DSN81010 . EMP	PLAN=AUD
2011-05-24 17:04:12.0	127.0.0.1	SC63	DB2R1	DB2R1	SELECT COUNT (*) INTO : H FROM DSN81010 . DEPT	PLAN=AUD
2011-05-24 17:04:12.0	127.0.0.1	SC63	DB2R1	DB2R1	SELECT COUNT (*) INTO : H FROM DSN81010 . DEPT	PLAN=AUD

Figure 15-20 Static COBOL sample application report output

15.3.4 InfoSphere Guardium Vulnerability Assessment reporting

In addition to providing a robust audit reporting and alert capability, InfoSphere Guardium provides some additional capability for further ensuring compliance and security requirements. Infosphere Guardium Database Vulnerability Assessment scans the database infrastructure for vulnerabilities and provides evaluation of database and data security health, with real-time and historical measurements:

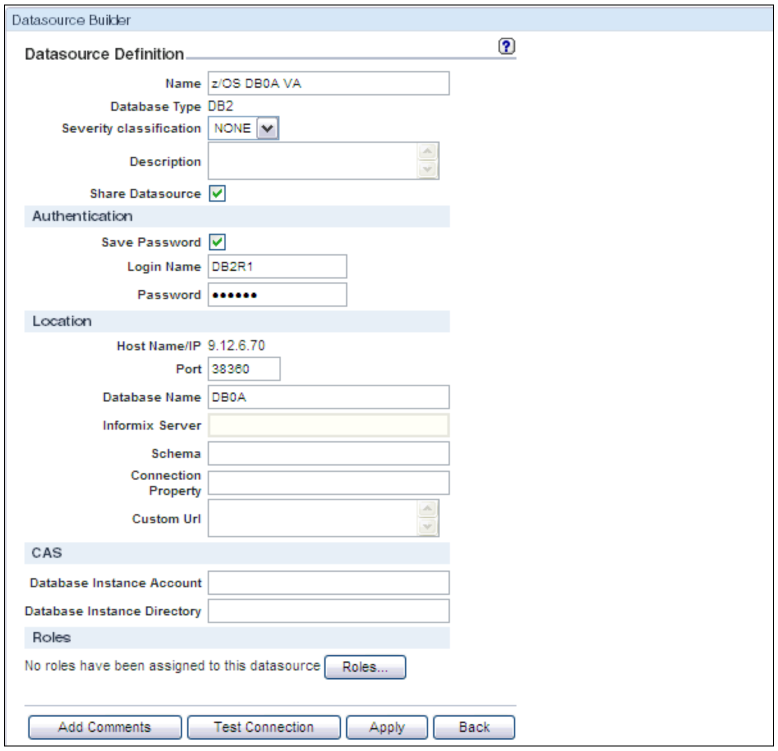
- Infosphere Guardium Database Vulnerability Assessment enables organizations to identify and address database vulnerabilities in a consistent and automated fashion.
- The assessment process evaluates the health of the database environment and recommends improvements by assessing the system configuration against best practices and finding vulnerabilities or potential threats to database resources, including configuration and behavioral risks.
- Infosphere Guardium Database Vulnerability Assessment recommend and prioritizes an action plan based on discovered areas of the most critical risks and vulnerabilities. The generation of reports and recommendations provide guidelines about how to meet compliance changes and elevate security of the evaluated database environment.

Infosphere Guardium Database Vulnerability Assessment connects directly to the database through the use of JDCB, and requires the installation of the z/OS License JAR file.

After the license file has been uploaded to the appliance, the next step is the creation of a connection definition, called a data source. Data sources are used by a number of applications and tools, such as Vulnerability Assessment and Classification. A data source identifies a specific database or file on a remote system. Data sources can be shared, but access is restricted according to the roles assigned to both the data source and the application that uses it.

Each data source is created for a type of application (for example, Classification). Different Infosphere Guardium applications require different types of database access. Data source definitions might require the storing of credentials to access the host system. For example, the Value Change Auditing application requires a high level of administrative access to the database, and it would not be appropriate to use that data source for other applications not requiring that level of privileges.

Figure 15-21 shows the data source definition used to connect to our DB2 for z/OS environment.



Name	z/OS DB0A VA
Database Type	DB2
Severity classification	NONE
Description	
Share Datasource	✓
Authentication	
Save Password	✓
Login Name	DB2R1
Password	••••••
Location	
Host Name/IP	9.12.6.70
Port	38360
Database Name	DB0A
Informix Server	
Schema	
Connection Property	
Custom Url	
CAS	
Database Instance Account	
Database Instance Directory	
Roles	

No roles have been assigned to this datasource

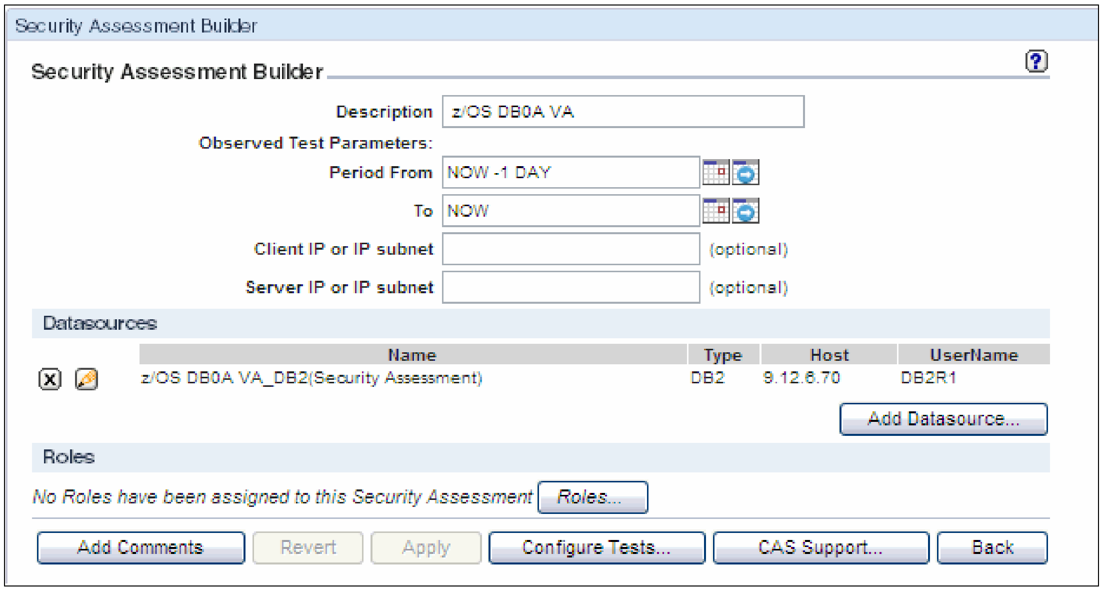
Figure 15-21 Data source definition for Vulnerability Assessment

The following elements are used in the creation of the definition:

- Login name: The RACF ID with connect privileges to the database server and select privileges against the DB2 catalog.
- Password*:* The password associated with this RACF ID.
- Host Name/IP: The IP address of the MVS LPAR hosting the DB2 subsystem.
- Port: The port number for the designated DB2 z/OS listener; by default, this is 446.
- Database Name: This is the DDF location name for the DB2 z/OS subsystem.

When the definition is completed, it can be validated by clicking the **Test Connection** button at the bottom of the form.

After the data source has been defined, the next step is to create the Vulnerability Assessment using the Security Assessment Builder, as shown in Figure 15-22.


Security Assessment Builder

Security Assessment Builder

Description	z/OS DB0A VA
Observed Test Parameters:	
Period From	NOW -1 DAY
To	NOW
Client IP or IP subnet	(optional)
Server IP or IP subnet	(optional)

Datasources

Name	Type	Host	UserName
z/OS DB0A VA_DB2(Security Assessment)	DB2	9.12.6.70	DB2R1

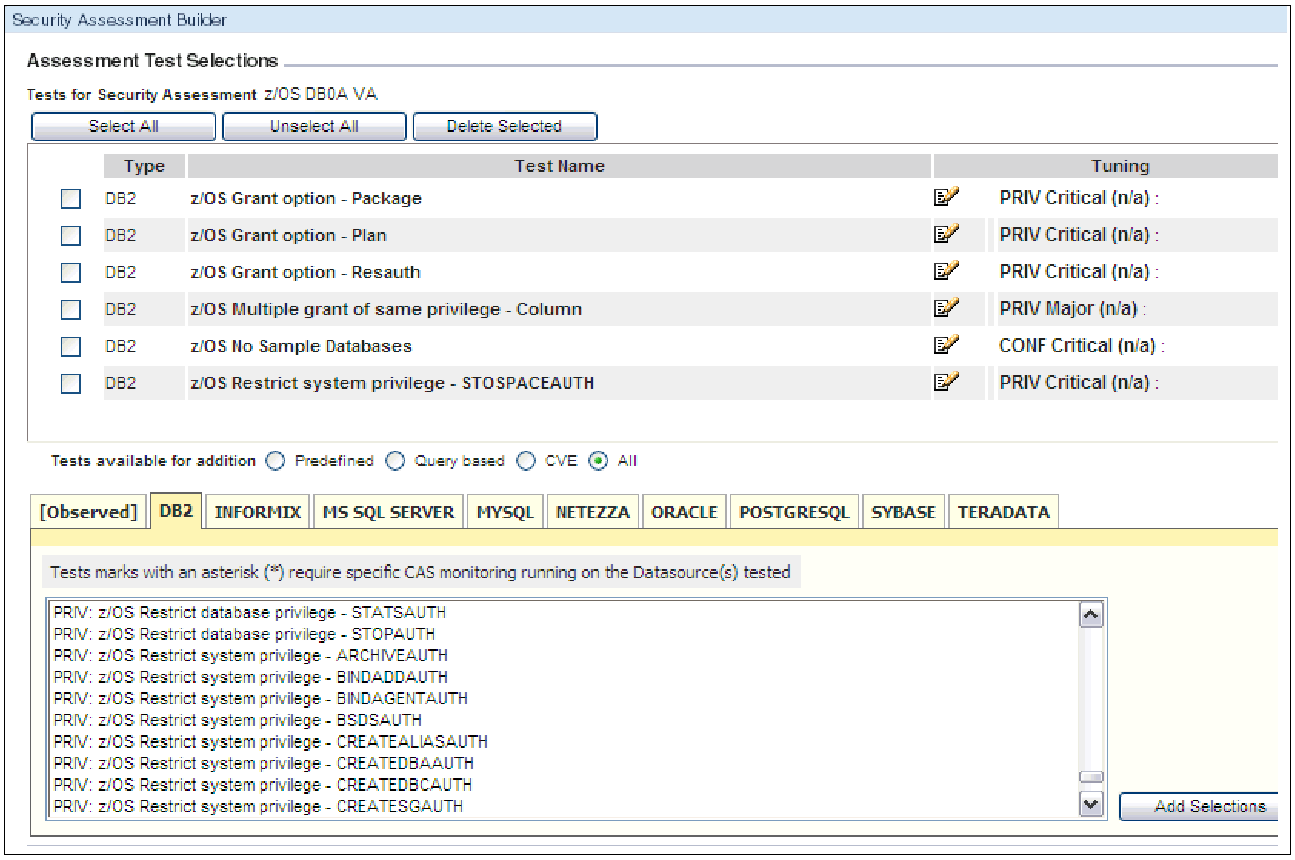
Roles

No Roles have been assigned to this Security Assessment

Figure 15-22 Security assessment builder

You assign the security assessment a name in the Description field. Click the **Add Datasource** button. A drop-down menu opens, where you select the data source created in the previous step. Then select the **Configure Test** radio button to present the list of assessment test categories.

Figure 15-23 shows the list of assessment test categories.



Security Assessment Builder

Assessment Test Selections

Tests for Security Assessment z/OS DB0A VA

Select All | Unselect All | Delete Selected

	Type	Test Name		Tuning
☐	DB2	z/OS Grant option - Package		PRIV Critical (n/a) :
☐	DB2	z/OS Grant option - Plan		PRIV Critical (n/a) :
☐	DB2	z/OS Grant option - Resauth		PRIV Critical (n/a) :
☐	DB2	z/OS Multiple grant of same privilege - Column		PRIV Major (n/a) :
☐	DB2	z/OS No Sample Databases		CONF Critical (n/a) :
☐	DB2	z/OS Restrict system privilege - STOSPACEAUTH		PRIV Critical (n/a) :

Tests available for addition ◯ Predefined ◯ Query based ◯ CVE [✓ All]

[Observed] | DB2 | INFORMIX | MS SQL SERVER | MYSQL | NETEZZA | ORACLE | POSTGRESQL | SYBASE | TERADATA

Tests marks with an asterisk (*) require specific CAS monitoring running on the Datasource(s) tested

```
PRIV: z/OS Restrict database privilege - STATSAUTH
PRIV: z/OS Restrict database privilege - STOPAUTH
PRIV: z/OS Restrict system privilege - ARCHIVEAUTH
PRIV: z/OS Restrict system privilege - BINDADDAUTH
PRIV: z/OS Restrict system privilege - BINDAGENTAUTH
PRIV: z/OS Restrict system privilege - BSDSAUTH
PRIV: z/OS Restrict system privilege - CREATEALIASAUTH
PRIV: z/OS Restrict system privilege - CREATEDBAAUTH
PRIV: z/OS Restrict system privilege - CREATEDBCAUTH
PRIV: z/OS Restrict system privilege - CREATESGAUTH
```

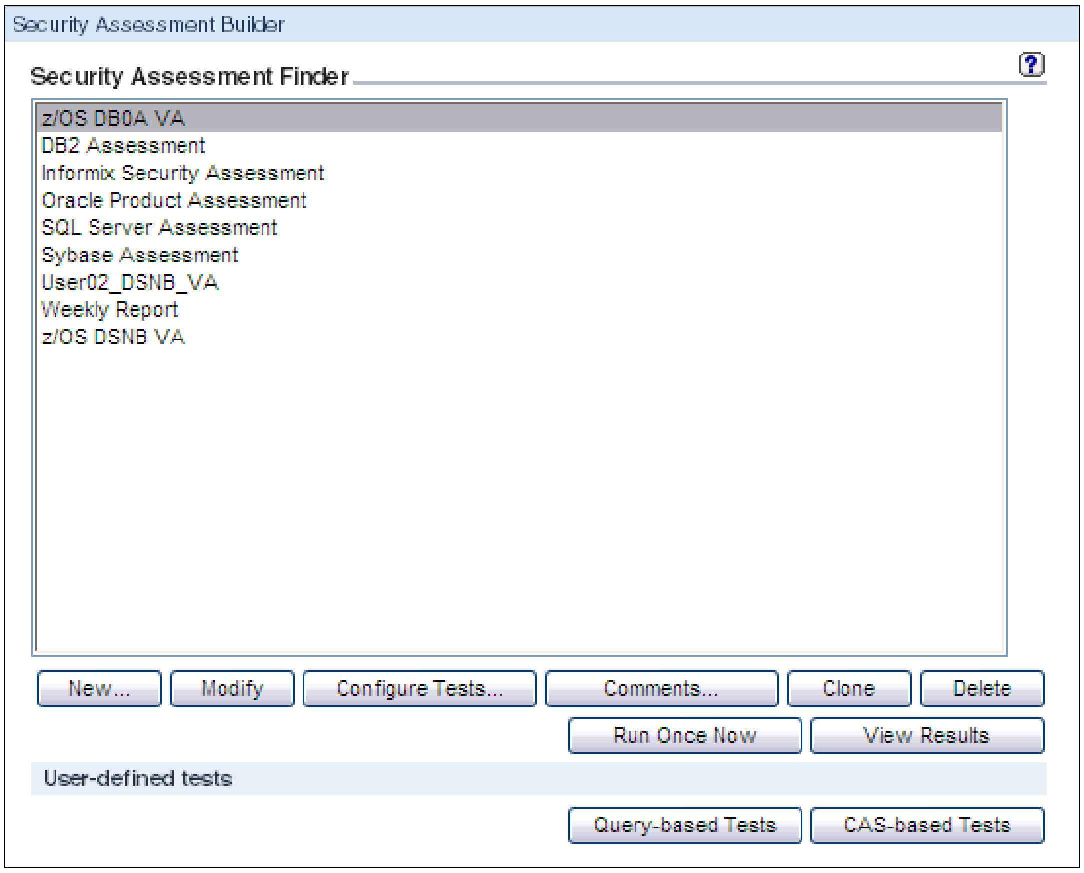
Add Selections

Figure 15-23 Vulnerability Assessment Test Selections

The different tests are categorized by type of DBMS. In Figure 15-23, we have highlighted the DB2 tab. The scroll bar allows you to scroll down the list to find the DB2 for z/OS tests. In this example, we have selected six different tests. The tuning category is used to prioritize the severity of the test when the vulnerability assessment report is generated. These criticality settings are the defaults, based on the recommendation of industry standard security assessment procedures, such as the DISA STIG and the CIS security standard. These severity settings can be customized by each customer.

After the assessment has been saved, it can then be located in the Security Assessment Finder window. Security assessment processes can be scheduled to execute on a predetermined schedule, or can submitted for immediate execution.

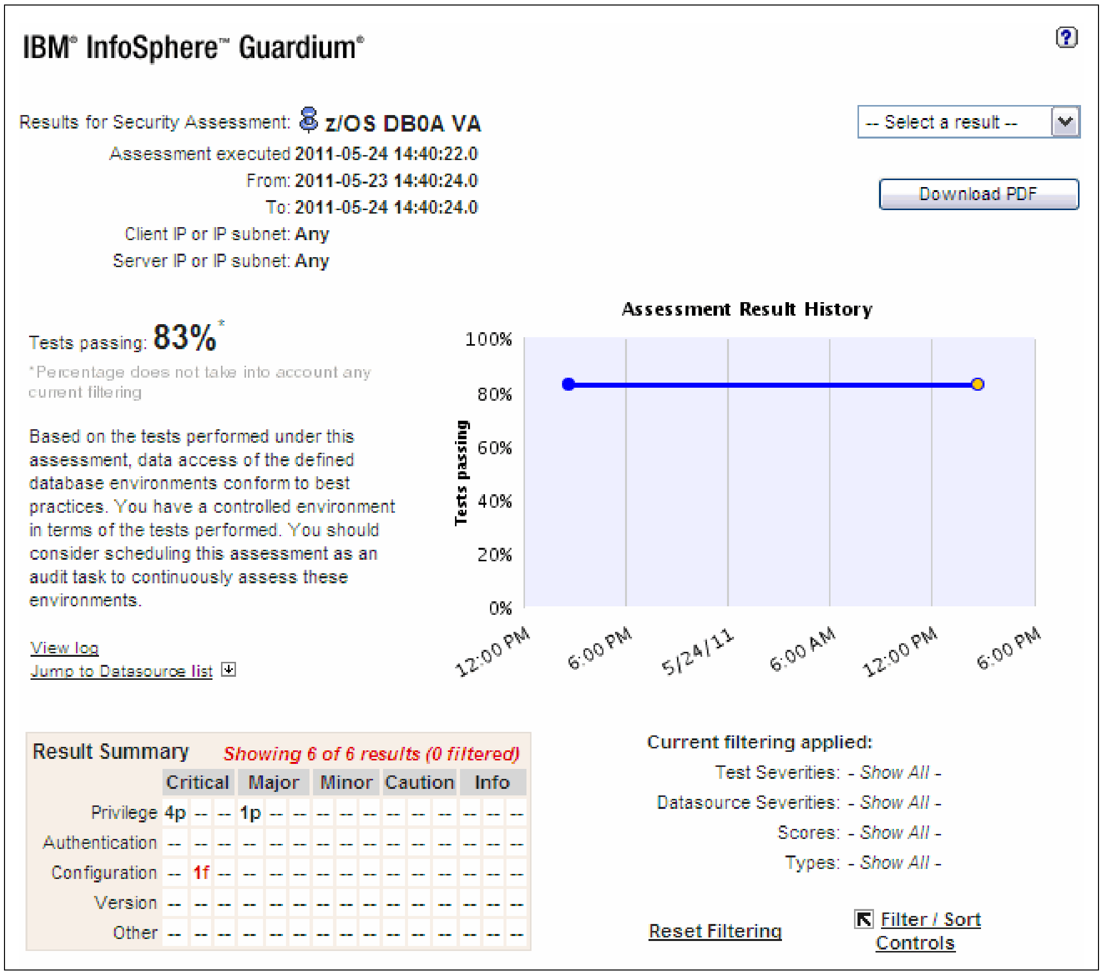
The Security Assessment Finder window is shown in Figure 15-24.



z/OS DB0A VA
DB2 Assessment
Informix Security Assessment
Oracle Product Assessment
SQL Server Assessment
Sybase Assessment
User02_DSNB_VA
Weekly Report
z/OS DSNB VA

Figure 15-24 Security Assessment Finder

Depending on the number of tests selected, and the size of the DB2 for z/OS catalog, the assessment might take a while to complete. When the execution is completed, clicking the **View Results** button displays the assessment report results. The first part of this report is shown in Figure 15-25.



Results for Security Assessment: z/OS DB0A VA

Assessment executed 2011-05-24 14:40:22.0

From: 2011-05-23 14:40:24.0

To: 2011-05-24 14:40:24.0

Client IP or IP subnet: Any

Server IP or IP subnet: Any

Tests passing: 83%*

*Percentage does not take into account any current filtering

Based on the tests performed under this assessment, data access of the defined database environments conform to best practices. You have a controlled environment in terms of the tests performed. You should consider scheduling this assessment as an audit task to continuously assess these environments.

Result Summary *Showing 6 of 6 results (0 filtered)*

	Critical			Major			Minor			Caution			Info		
Privilege	4p	--	--	1p	--	--	--	--	--	--	--	--	--	--	--
Authentication	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Configuration	--	1f	--	--	--	--	--	--	--	--	--	--	--	--	--
Version	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Other	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Current filtering applied:

Test Severities: - *Show All* -

Datasource Severities: - *Show All* -

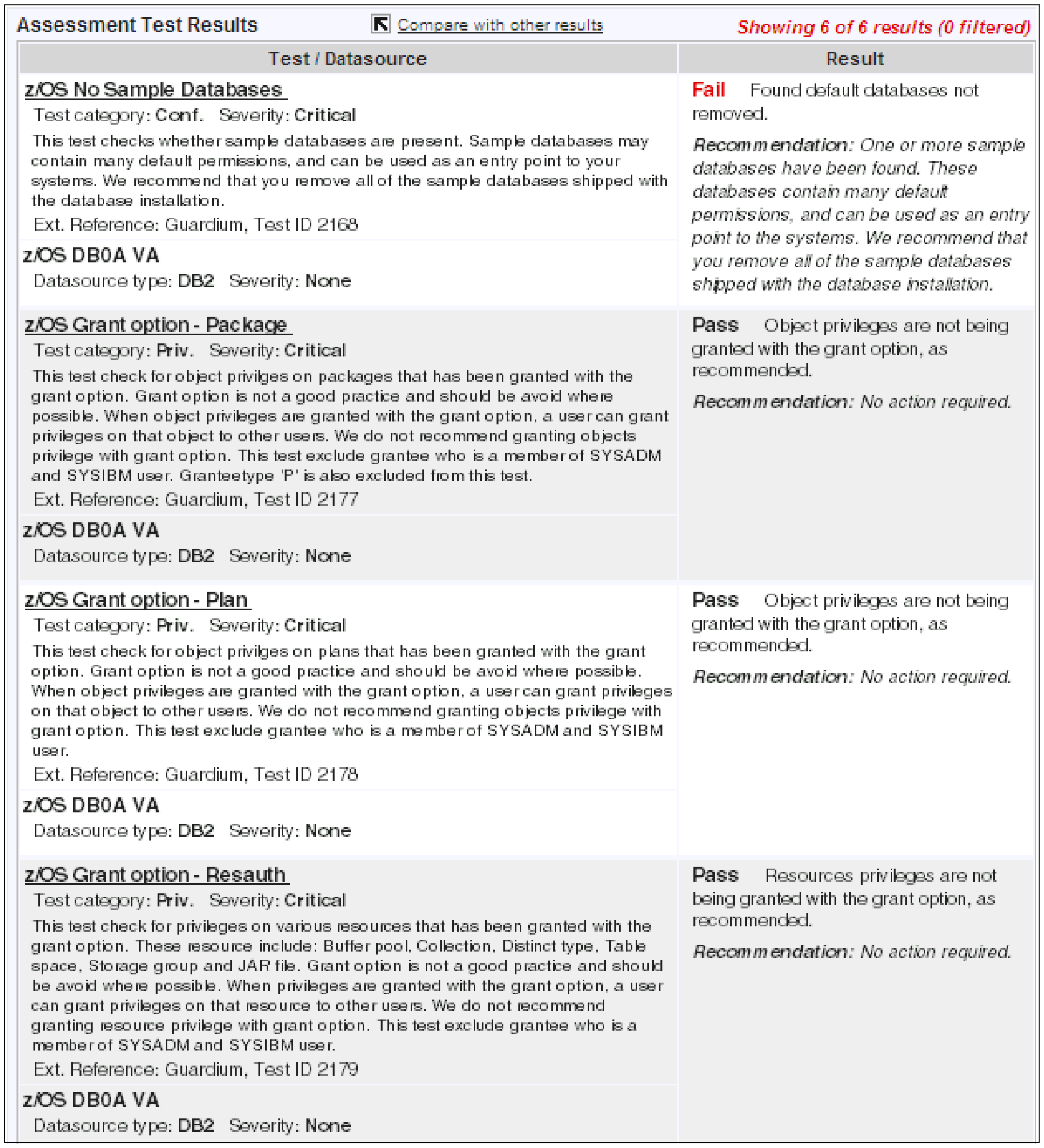
Scores: - *Show All* -

Types: - *Show All* -

Figure 15-25 Vulnerability assessment report - Part 1

This section of the report provides a summary of the vulnerability assessment results. The results are summarized by category of test and whether the test resulted in a pass or fail. In addition, the assessment results are stored in a historical table, and every time the assessment is executed, the assessment results history shows the histogram of results over time. This is a valuable view of the assessment results, as it provides the ability to demonstrate historical improvements to the security environment over time.

Figure 15-26 shows an example of the detail report section.



Assessment Test Results   Compare with other results   Showing 6 of 6 results (0 filtered)

Test / Datasource	Result
z/OS No Sample Databases Test category: Conf. Severity: Critical This test checks whether sample databases are present. Sample databases may contain many default permissions, and can be used as an entry point to your systems. We recommend that you remove all of the sample databases shipped with the database installation. Ext. Reference: Guardium, Test ID 2168 z/OS DB0A VA Datasource type: DB2 Severity: None	Fail Found default databases not removed. *Recommendation: One or more sample databases have been found. These databases contain many default permissions, and can be used as an entry point to the systems. We recommend that you remove all of the sample databases shipped with the database installation.*
z/OS Grant option - Package Test category: Priv. Severity: Critical This test check for object privilges on packages that has been granted with the grant option. Grant option is not a good practice and should be avoid where possible. When object privileges are granted with the grant option, a user can grant privileges on that object to other users. We do not recommend granting objects privilege with grant option. This test exclude grantee who is a member of SYSADM and SYSIBM user. Granteetype 'P' is also excluded from this test. Ext. Reference: Guardium, Test ID 2177 z/OS DB0A VA Datasource type: DB2 Severity: None	Pass Object privileges are not being granted with the grant option, as recommended. *Recommendation: No action required.*
z/OS Grant option - Plan Test category: Priv. Severity: Critical This test check for object privilges on plans that has been granted with the grant option. Grant option is not a good practice and should be avoid where possible. When object privileges are granted with the grant option, a user can grant privileges on that object to other users. We do not recommend granting objects privilege with grant option. This test exclude grantee who is a member of SYSADM and SYSIBM user. Ext. Reference: Guardium, Test ID 2178 z/OS DB0A VA Datasource type: DB2 Severity: None	Pass Object privileges are not being granted with the grant option, as recommended. *Recommendation: No action required.*
z/OS Grant option - Resauth Test category: Priv. Severity: Critical This test check for privileges on various resources that has been granted with the grant option. These resource include: Buffer pool, Collection, Distinct type, Table space, Storage group and JAR file. Grant option is not a good practice and should be avoid where possible. When privileges are granted with the grant option, a user can grant privileges on that resource to other users. We do not recommend granting resource privilege with grant option. This test exclude grantee who is a member of SYSADM and SYSIBM user. Ext. Reference: Guardium, Test ID 2179 z/OS DB0A VA Datasource type: DB2 Severity: None	Pass Resources privileges are not being granted with the grant option, as recommended. *Recommendation: No action required.*

Figure 15-26 Vulnerability assessment report - Part 2

For each test, a description of what vulnerability the test covers, the reason for the score associated with the test, and, for tests that result in fail, a recommendation for remediation. If desired, the vulnerability assessment report can be converted into a PDF and downloaded for distribution.

15.3.5 InfoSphere Guardium data classification

For more information about the requirements for data discovery, refer to 2.1, “DB2 and z/OS threat environment” on page 20.

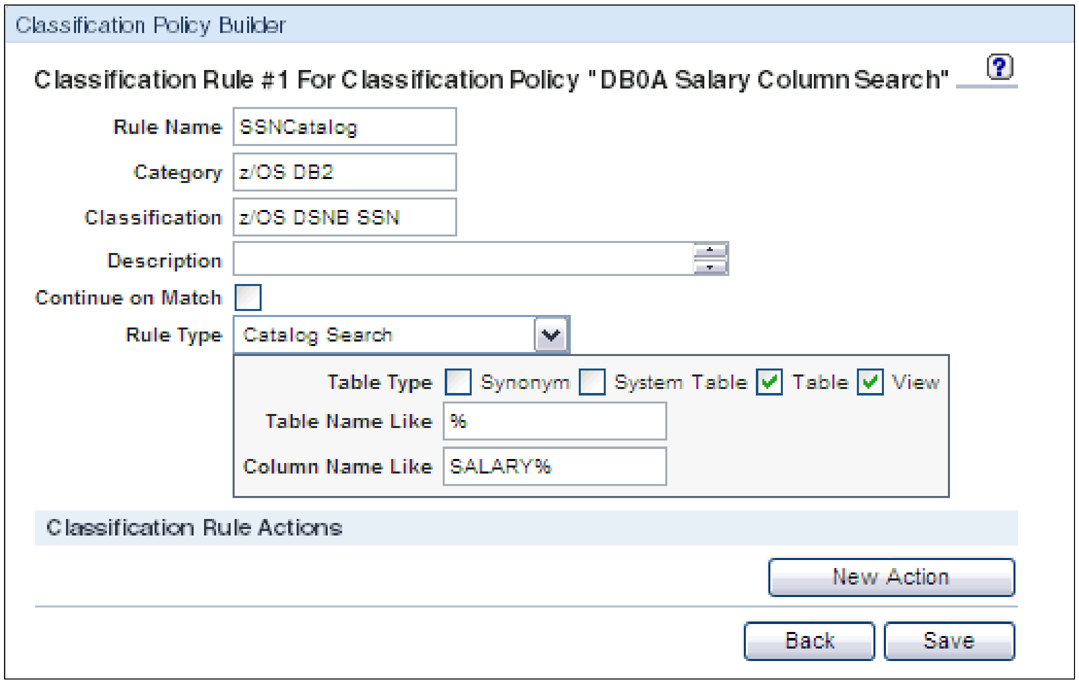
Infosphere Guardium provides the Classification feature to discover and classify sensitive data, so that you can make and enforce effective access policy decisions.

A classification policy is a set of rules designed to discover and tag sensitive data elements (database tables or files). You can define a set of actions to be taken for each rule. An action might be to generate an email alert, or to add a member to a (Infosphere Guardium) group. Each time a rule is satisfied, that event is logged, and thus can be reported upon (unless Ignore is specified as the action to be taken, in which case there is no logging for that rule).

A data source, described in 15.3.4, "InfoSphere Guardium Vulnerability Assessment reporting" on page 373, identifies a specific database instance, and its definition on the Infosphere Guardium appliance can optionally store account information and any other parameters required to access the database. Data source definitions can be shared, and can be used by other applications in addition to the classification function.

Classification using DB2 catalog information

For our environment, we first defined a classification policy to search the catalog on our DB2 for z/OS subsystem to look for tables that contain columns named SALARY. The first step in the definition process is to define the policy using the Policy Editor window. Figure 15-27 shows a representative example of this window.



Classification Rule #1 For Classification Policy "DB0A Salary Column Search"

Rule Name	SSNCatalog
Category	z/OS DB2
Classification	z/OS DSNB SSN
Description	
Continue on Match	☐
Rule Type	Catalog Search
Table Type	☐ Synonym ☐ System Table ✓ Table ✓ View
Table Name Like	%
Column Name Like	SALARY%

Figure 15-27 Classification rule for catalog search

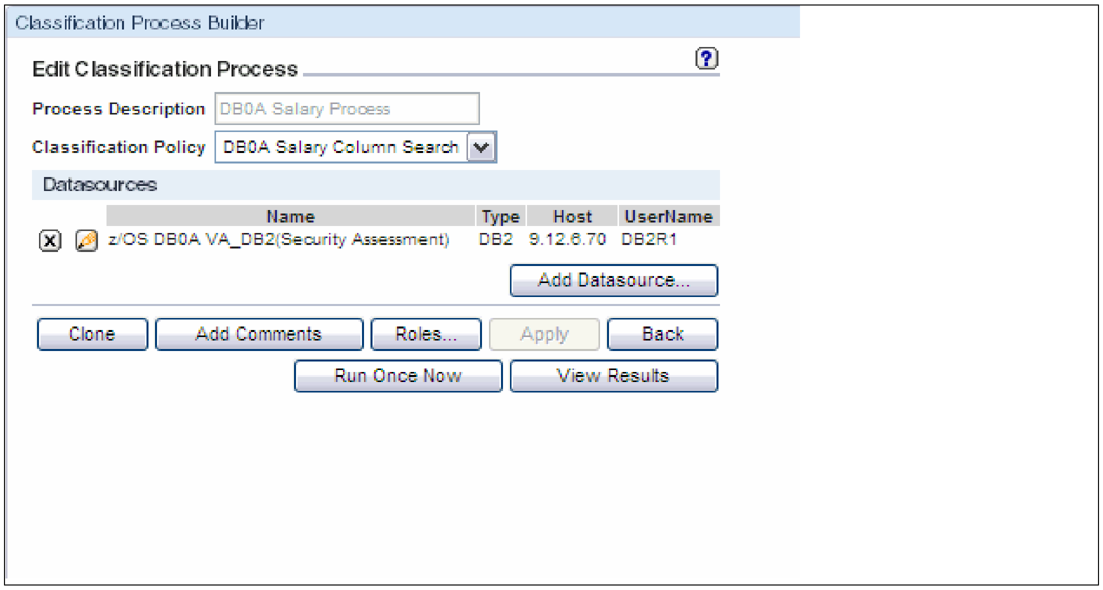
When specifying the policy, the following fields are needed:

- Table Type: For our scenario, we chose Table and View.
- Table Name: In large catalog environments, you can narrow the search down to limit catalog search tim. In our environment, we chose to search all table entries.
- Column Name Like: This field defines what column name in the catalog name to search on,. For our scenario, we selected SALARY.

When defining classification rules, you can attach actions to the process, which might include generating an alert, building a report, or adding the tables to a group, which can then be used for other Infosphere Guardium processes. In our scenario, we just run the classification without any attached action.

After the policy has been defined, it needs to be associated with a classification process. The definition of the classification process is what ties the classification policy to one or more data sources.

A classification process defines a job consisting of a classification policy and one or more data sources. Any classification process can be run on an ad hoc basis. If a data source referenced by that process has not stored login information, you are prompted to supply the necessary login parameters. If login information has been stored for all the data sources used in a classification process, that process can be included as a classification task in a compliance workflow automation process, which can be run on an on demand or scheduled basis. Figure 15-28 shows a representative view of the classification process window.



Process Description: DB0A Salary Process
Classification Policy: DB0A Salary Column Search

Name	Type	Host	UserName
z/OS DB0A VA_DB2(Security Assessment)	DB2	9.12.6.70	DB2R1

Figure 15-28 Classification process builder

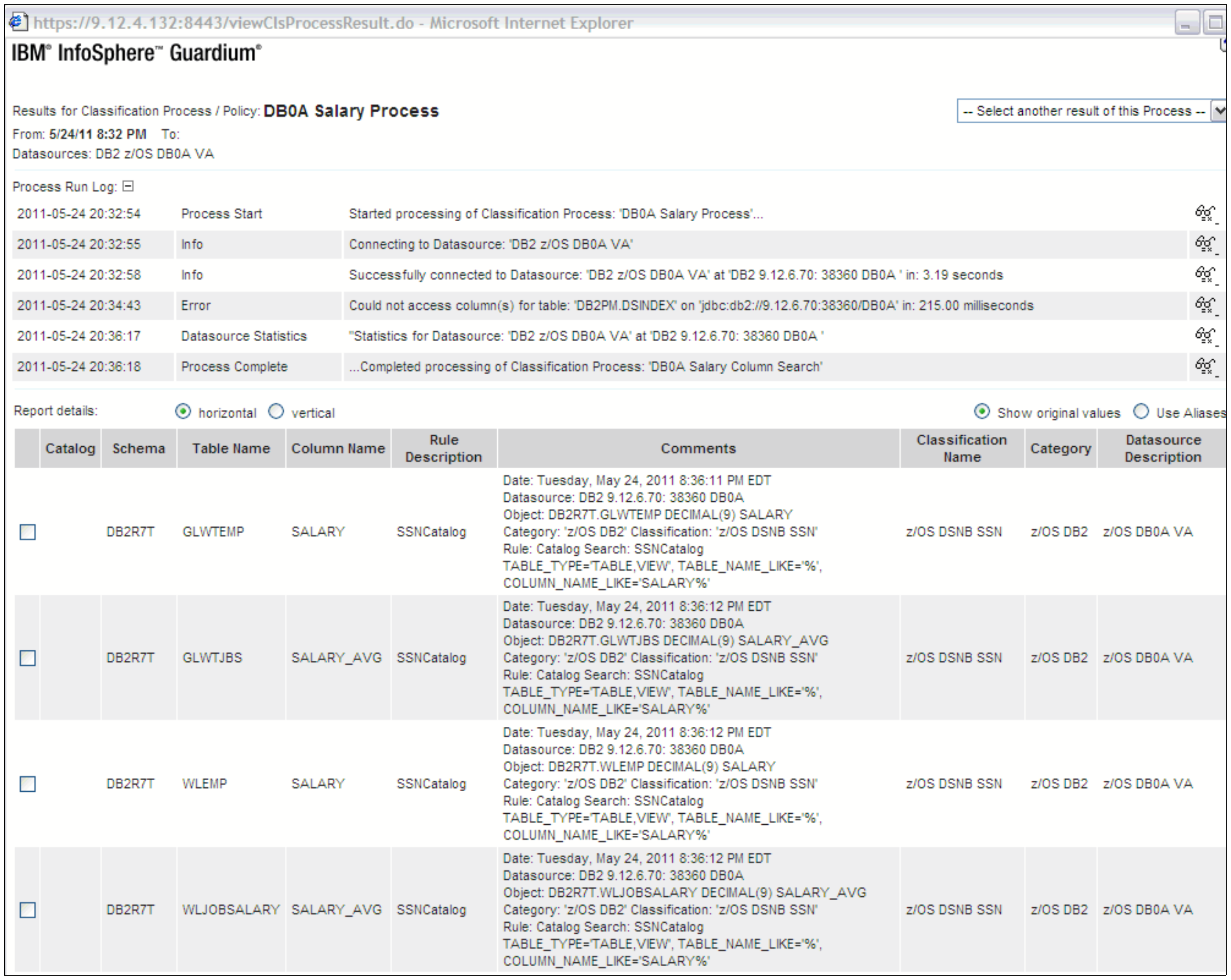
Clicking the **Add Datasource** button opens a drop-down menu that lists all the defined data source definitions, and the data source is picked from this list. Classification processes can be scheduled to run on a predefined basis, or can be scheduled as ad hoc processes, by clicking the **Run Once Now** button.

In general, if there are many rules that are sampling data, the load on the database server should remain constant, but the process may take additional time to run.The Classifier also throttles itself to be periodically idle so that it does not overwhelm the database server with requests.

If any one query takes longer than 12 minutes, the query is cancelled, a message is logged, and no more data is sampled for that table. If any rows were acquired while sampling, they are used to evaluate the rule for that table. This action usually only happens on servers that are experiencing performance problems in general.

If, for some reason, an operation on the Processor takes over approximately 30 minutes, the entire process is halted, a message is logged with the process statistics, and the next Classification Process is started.

After the process completes, the resultant report shown in Figure 15-29 displays.



https://9.12.4.132:8443/viewClsProcessResult.do - Microsoft Internet Explorer

IBM® InfoSphere™ Guardium®

Results for Classification Process / Policy: **DB0A Salary Process**
From: **5/24/11 8:32 PM** To:
Datasources: DB2 z/OS DB0A VA

Process Run Log:

```
2011-05-24 20:32:54   Process Start          Started processing of Classification Process: 'DB0A Salary Process'...
2011-05-24 20:32:55   Info                   Connecting to Datasource: 'DB2 z/OS DB0A VA'
2011-05-24 20:32:58   Info                   Successfully connected to Datasource: 'DB2 z/OS DB0A VA' at 'DB2 9.12.6.70: 38360 DB0A ' in: 3.19 seconds
2011-05-24 20:34:43   Error                  Could not access column(s) for table: 'DB2PM.DSINDEX' on 'jdbc:db2://9.12.6.70:38360/DB0A' in: 215.00 milliseconds
2011-05-24 20:36:17   Datasource Statistics  "Statistics for Datasource: 'DB2 z/OS DB0A VA' at 'DB2 9.12.6.70: 38360 DB0A '
2011-05-24 20:36:18   Process Complete       ...Completed processing of Classification Process: 'DB0A Salary Column Search'
```

Report details: [✓ horizontal] vertical   [✓ Show original values] Use Aliases

	Catalog	Schema	Table Name	Column Name	Rule Description	Comments	Classification Name	Category	Datasource Description
☐		DB2R7T	GLWTEMP	SALARY	SSNCatalog	Date: Tuesday, May 24, 2011 8:36:11 PM EDT Datasource: DB2 9.12.6.70: 38360 DB0A Object: DB2R7T.GLWTEMP DECIMAL(9) SALARY Category: 'z/OS DB2' Classification: 'z/OS DSNB SSN' Rule: Catalog Search: SSNCatalog TABLE_TYPE='TABLE,VIEW', TABLE_NAME_LIKE='%', COLUMN_NAME_LIKE='SALARY%'	z/OS DSNB SSN	z/OS DB2	z/OS DB0A VA
☐		DB2R7T	GLWTJBS	SALARY_AVG	SSNCatalog	Date: Tuesday, May 24, 2011 8:36:12 PM EDT Datasource: DB2 9.12.6.70: 38360 DB0A Object: DB2R7T.GLWTJBS DECIMAL(9) SALARY_AVG Category: 'z/OS DB2' Classification: 'z/OS DSNB SSN' Rule: Catalog Search: SSNCatalog TABLE_TYPE='TABLE,VIEW', TABLE_NAME_LIKE='%', COLUMN_NAME_LIKE='SALARY%'	z/OS DSNB SSN	z/OS DB2	z/OS DB0A VA
☐		DB2R7T	WLEMP	SALARY	SSNCatalog	Date: Tuesday, May 24, 2011 8:36:12 PM EDT Datasource: DB2 9.12.6.70: 38360 DB0A Object: DB2R7T.WLEMP DECIMAL(9) SALARY Category: 'z/OS DB2' Classification: 'z/OS DSNB SSN' Rule: Catalog Search: SSNCatalog TABLE_TYPE='TABLE,VIEW', TABLE_NAME_LIKE='%', COLUMN_NAME_LIKE='SALARY%'	z/OS DSNB SSN	z/OS DB2	z/OS DB0A VA
☐		DB2R7T	WLJOBSALARY	SALARY_AVG	SSNCatalog	Date: Tuesday, May 24, 2011 8:36:12 PM EDT Datasource: DB2 9.12.6.70: 38360 DB0A Object: DB2R7T.WLJOBSALARY DECIMAL(9) SALARY_AVG Category: 'z/OS DB2' Classification: 'z/OS DSNB SSN' Rule: Catalog Search: SSNCatalog TABLE_TYPE='TABLE,VIEW', TABLE_NAME_LIKE='%', COLUMN_NAME_LIKE='SALARY%'	z/OS DSNB SSN	z/OS DB2	z/OS DB0A VA

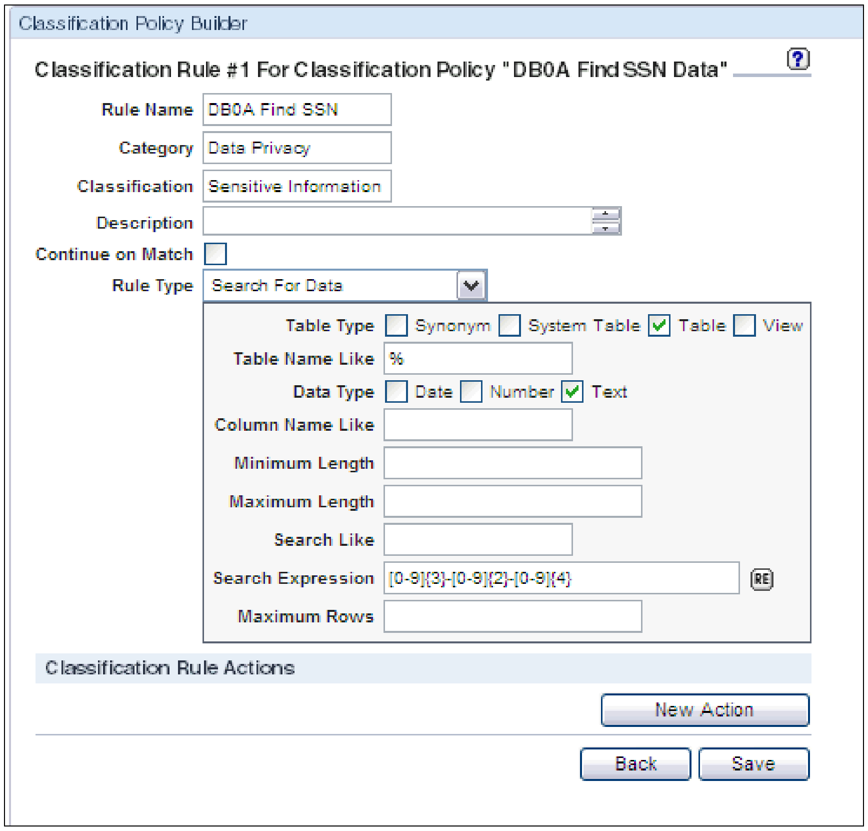
Figure 15-29 Catalog classification process execution report

If we wanted to add one or more results from the search into a group, we can select the entry and add it to an existing group on an ad hoc basis. Also, notice that when conducting the catalog search, there was a lock on a catalog entry, which was noted and then bypassed.

Classification based on data patterns

The previous scenario was based on the assumption that one or more columns names would describe sensitive data. Unfortunately, in many cases, the table or column names are not completely reflective of the sensitive nature of the data stored inside of them. In this scenario, we show how to use Infosphere Guardium to search for sensitive data using pattern matching techniques.

As described in “Classification using DB2 catalog information” on page 380, we use the existing data source. We then create a new classification policy, similar to the action previously described, as shown in Figure 15-30.



Classification Rule #1 For Classification Policy "DB0A Find SSN Data"

Rule Name	DB0A Find SSN
Category	Data Privacy
Classification	Sensitive Information
Description	
Continue on Match	☐
Rule Type	Search For Data
Table Type	☐ Synonym ☐ System Table ✓ Table ☐ View
Table Name Like	%
Data Type	☐ Date ☐ Number ✓ Text
Column Name Like	
Minimum Length	
Maximum Length	
Search Like	
Search Expression	[0-9]{3}-[0-9]{2}-[0-9]{4}
Maximum Rows	

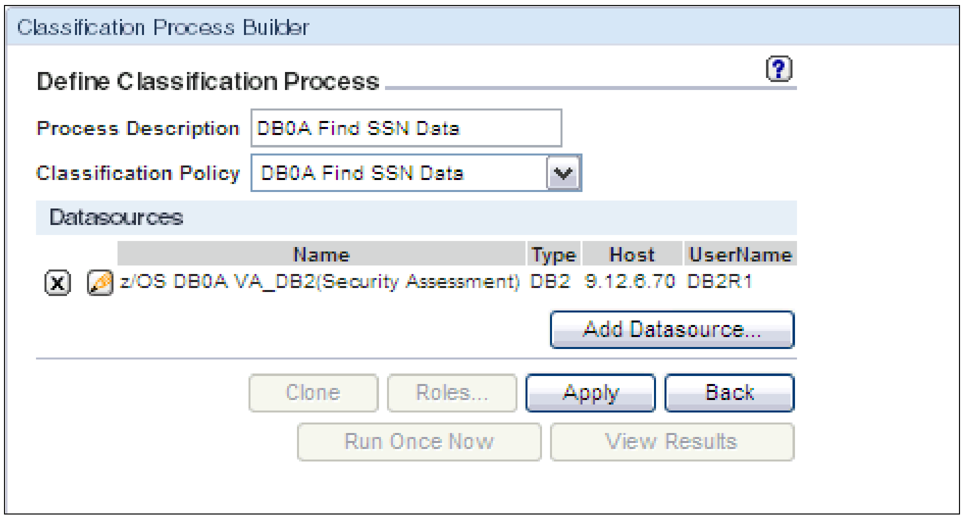
Figure 15-30 Classification rule for data pattern search

For this rule, we select the **Rule type** drop-down menu and select **Search for Data**. In addition, we specify a *search expression* to be used in this rule. This specific expression looks for:

- Characters 0 - 9 in the first three positions
- The character ‘-’ in position 4
- Characters 0 - 9 in positions 5 and 6
- The character ‘-’ in position 7
- Characters 0-9 in positions 8 - 11.

The social security number (SSN) is shown as 111-11-1111, which should match the regular expression coded in the classification rule.

The classification rule is then connected to a data source by the creation of the classification process, as shown in Figure 15-31.


Classification Process Builder

Define Classification Process

Process Description: DB0A Find SSN Data

Classification Policy: DB0A Find SSN Data

Datasources

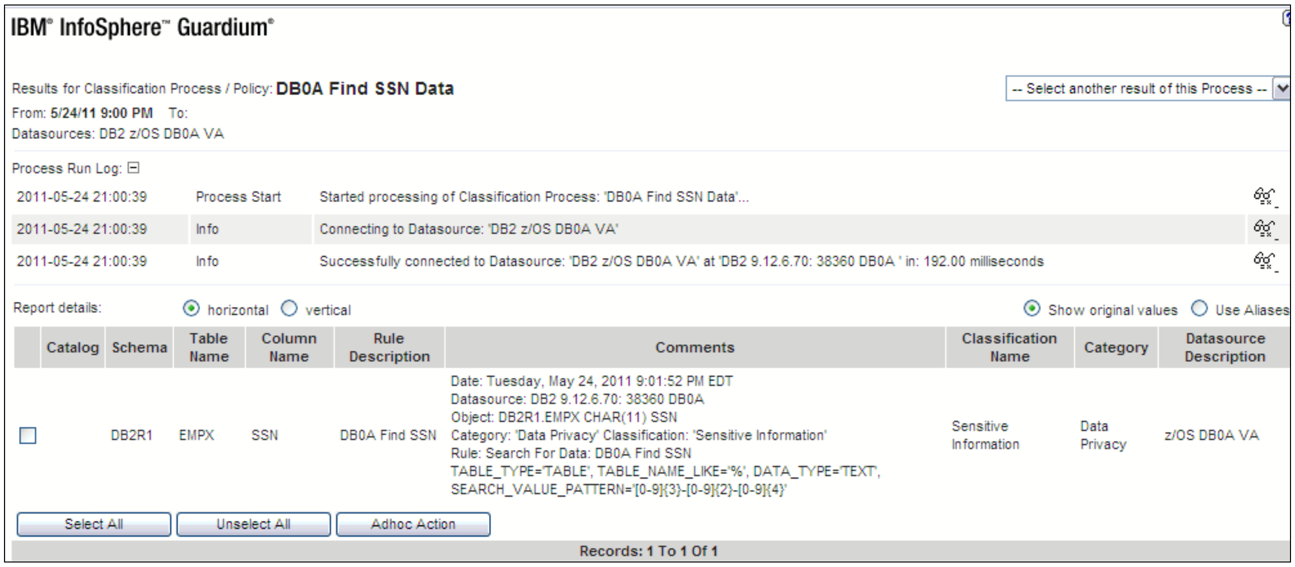
Name	Type	Host	UserName
z/OS DB0A VA_DB2(Security Assessment)	DB2	9.12.6.70	DB2R1

Figure 15-31 Classification process definition for SSN search

Once defined, the process can be scheduled, or run manually by selecting the **Run Once Now** button. As you might expect, to search all tables in a DB2 subsystem looking for patterns in the data might be resource intensive. To reduce the impact on system performance, the classification process works in the following manner.

To prevent overloading the server when scanning data, the classification process always samples the database, never reading more than 1,000 rows. It begins by scanning sets of 50 consecutive rows returned by the database server, beginning with the first row. The second set of 50 begins with the 1000th row. Thereafter, it skips ahead by powers or two, such that the next block of 50 begins 2 K, 4 K, 8 K, 16 K, 32 K, and so on. During this process, if any query takes longer than 10 seconds, the skip interval is multiplied by 10, so if the current sequence is 640 K, the next is 6.4 M, and so on (until 1,000 rows have been sampled or there are no more rows in the table). If the row limit on the Search for Data rule is set higher than 1000, the same sampling technique is used. Unless the table is quite large (greater than 262,144,000 rows), the sampling process runs out of rows before the maximum row limit is exceeded.

Figure 15-32 shows the results from the classification policy execution.


IBM® InfoSphere™ Guardium®

Results for Classification Process / Policy: **DB0A Find SSN Data**

From: **5/24/11 9:00 PM** To:

Datasources: DB2 z/OS DB0A VA

-- Select another result of this Process --

Process Run Log:

```
2011-05-24 21:00:39   Process Start   Started processing of Classification Process: 'DB0A Find SSN Data'...
2011-05-24 21:00:39   Info            Connecting to Datasource: 'DB2 z/OS DB0A VA'
2011-05-24 21:00:39   Info            Successfully connected to Datasource: 'DB2 z/OS DB0A VA' at 'DB2 9.12.6.70: 38360 DB0A ' in: 192.00 milliseconds
```

Report details: [✓ horizontal] vertical [✓ Show original values] Use Aliases

	Catalog	Schema	Table Name	Column Name	Rule Description	Comments	Classification Name	Category	Datasource Description
☐		DB2R1	EMPX	SSN	DB0A Find SSN	Date: Tuesday, May 24, 2011 9:01:52 PM EDT Datasource: DB2 9.12.6.70: 38360 DB0A Object: DB2R1.EMPX CHAR(11) SSN Category: 'Data Privacy' Classification: 'Sensitive Information' Rule: Search For Data: DB0A Find SSN TABLE_TYPE='TABLE', TABLE_NAME_LIKE='%', DATA_TYPE='TEXT', SEARCH_VALUE_PATTERN='[0-9]{3}-[0-9]{2}-[0-9]{4}'	Sensitive Information	Data Privacy	z/OS DB0A VA

Records: 1 To 1 Of 1

Figure 15-32 SSN data classification policy execution report

In our scenario, we only had one table that contained SSN information, which was DB2R1.EMPX, as shown in the report.

Part 5

Appendixes

This part includes the following appendixes:

A

Spiffy Computer Company security setup

This appendix describes the security setup for Spiffy Computer Company, the fictitious company we used in our scenarios.

A.1 Spiffy Computer Company

The mission of Spiffy Computer Company is to develop software for use on z/OS.

This scenario describes how to implement a security plan for Spiffy Computer Company. We will use the new DB2 10 authorities. We show how to monitor data access, including the new administrative authorities.

You should base your security plan, techniques, and procedures on your actual security objectives; do not use this sample security plan as an exact model for your security needs. Instead, use it to understand various possibilities and address problem areas that you might encounter when you implement your security plan.

A.1.1 Determining security objectives

An important step in defining and implementing an effective security plan is to determine your security objectives.

Here are the security objectives of the Spiffy Computer Company:

- Separate duties rather than maintain powerful authorities.
- Monitor access to tables holding sensitive data.
- Audit the use of authorities.
- Mask columns containing personally identifiable information.
- Easily manage access to sensitive rows without using views.

A.2 Organization of Spiffy Computer Company

There are seven large departments in Spiffy Computer Company, each with many teams, and each team has a manager and many employees. Figure A-1 shows the Spiffy Computer Company's organization chart.

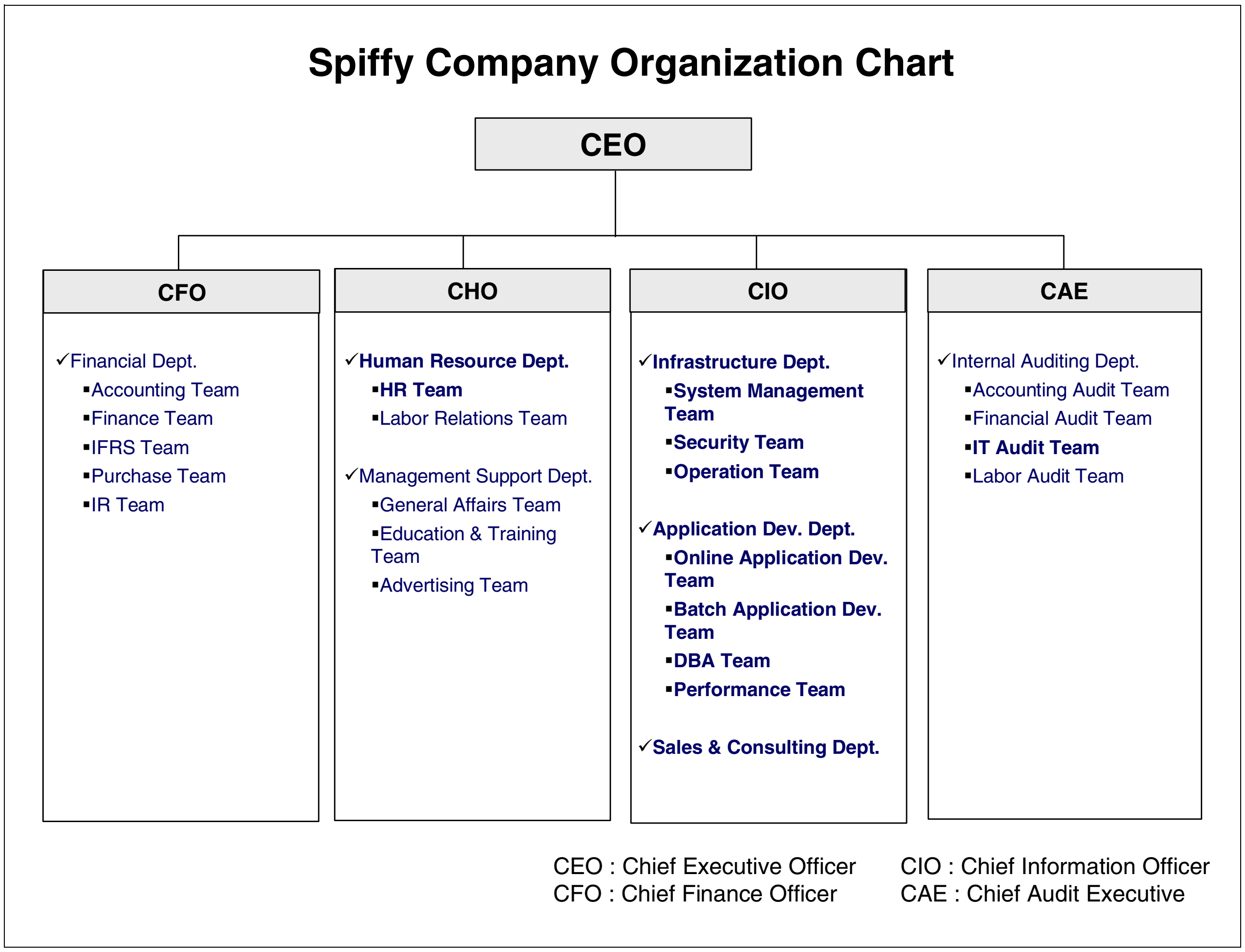


Figure A-1 Organization chart of Spiffy Computer Company

In our scenario, we mainly discuss the HR Team and Infrastructure Department.

The HR Team is responsible for personnel affairs, so much employees personal data is handled by them. Each team's manager can access some of an employee's personal data, but not all of it. The HR Team works only at the office.

The Infrastructure Department is responsible for managing and operating the production environment, making certain that it is available 24 x 7, and protecting the environment against outside or inside invaders (this last criteria is usually a joint responsibility of the Security Team and IT Audit Team). The System Management Team and Security Team work at the office and the Operation Team works at the data center.

The Application Development Department is responsible for developing applications based on the customer's requirements and also management business data and application performance. The Application Development Department also works only at the office. The DBA Team can work at home sometimes, but they work mostly at the office.

The Sales & Consulting Department is responsible for developing new businesses and contracting new orders, such as consulting services, and project and service outsourcing. They work in many different places (usually in the field).

B

Introduction to cryptography

In this appendix, we describe the fundamentals of cryptography and the Integrated Cryptographic Service Facility (ICSF) for z/OS.

This appendix contains the following topics:

- Fundamentals of cryptography
- Integrated Cryptographic Service Facility for z/OS
- Tivoli Key Lifecycle Manager

B.1 Fundamentals of cryptography

The requirement to encrypt data at rest is fundamental to many regulatory compliance initiatives. Encryption is the process where clear (or plain) text data is transformed into ciphertext. This transformation uses a mathematical formula, known as an encryption algorithm, in conjunction with a data encrypting key to create the ciphertext.

There are two basic techniques for encrypting information:

- Symmetric encryption, also called secret key encryption
- Asymmetric encryption, also known as public key encryption.

Symmetric encryption is the oldest approach and is the technique used by both the DB2 built-in function and the InfoSphere Guardium Data Encryption for DB2 and IMS Databases. A hexadecimal key, which is a string of randomly generated hexadecimal characters, is applied to the text of a message to change the content in a particular way, known as the encryption algorithm. As long as both sender and recipient know the secret key, and use the same encryption algorithm, they can encrypt and decrypt all the data encrypted by that key.

The problem with secret keys is exchanging them over the Internet or a large network while preventing them from falling into the wrong hands. Anyone who knows the secret key can decrypt the message. One answer is asymmetric encryption, in which there are two related keys, that is, a key pair. A public key is made freely available to anyone who might want to send you a message. A second, private key is kept secret, so that only you know it.

Public / private key algorithms are used to protect (encrypt) a symmetric (TDES/AES) key. That symmetric key is used to encrypt the data. The private key is used to recover the symmetric key. This process is also how SSL/TLS/HTTPS works for network encryption.

Asymmetric encryption, along with the use of digital certificates, is mostly seen in the network encryption realm; a common example is the HTTPS implementation in web applications.

Starting with the original clear text data and using the chosen encryption algorithm or mathematical formula, which applies the key to the clear text portion, the clear text is encrypted and becomes ciphertext. When viewed, ciphertext appears to be set of randomly generated hexadecimal values. When an application request requires clear text data, the same encryption algorithm and key value are applied to the ciphertext, and the original clear text data values appear.

The process of creating ciphertext is also referred to as encryption, and the process of turning ciphertext to clear text is known as decryption. Figure B-1 shows a graphical representation of this process.

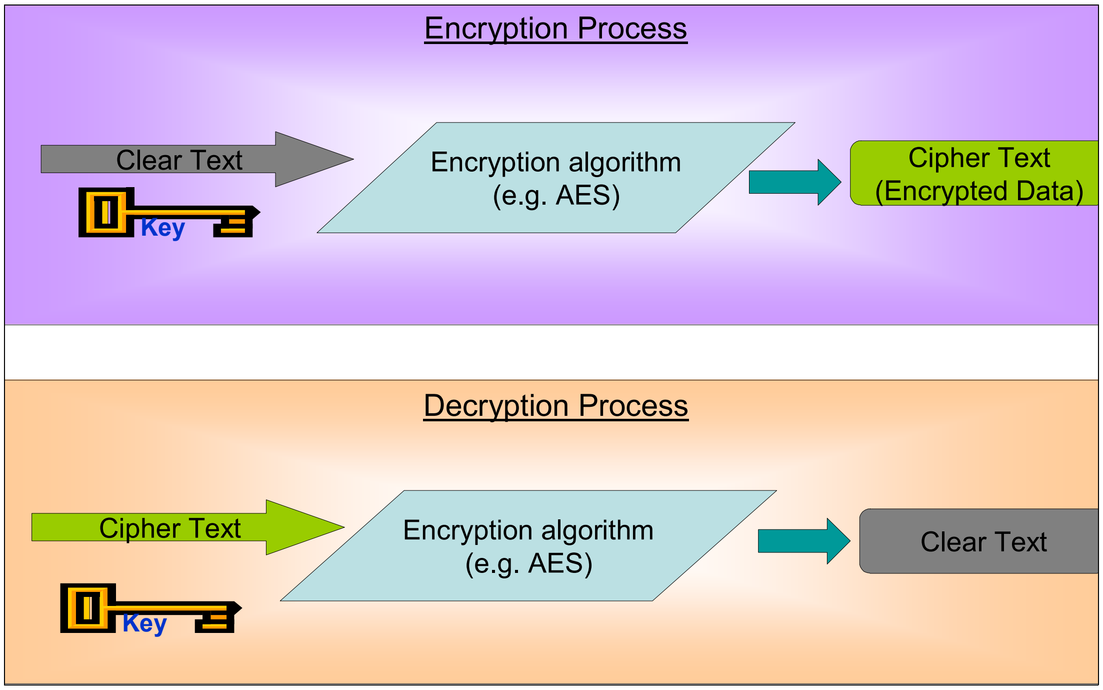

Encryption Process
Clear Text
Key
Encryption algorithm (e.g. AES)
Cipher Text (Encrypted Data)
Decryption Process
Cipher Text
Key
Encryption algorithm (e.g. AES)
Clear Text

Figure B-1 Representation of encryption and decryption

B.1.1 Encryption algorithms

There are two different encryption algorithms that are part of the hardware encryption support on System z. These algorithms are the Triple Data Encryption Standard (TDES) and the American Encryption Standard (AES). Both are encryption algorithms that are currently considered secure by industry recognized standards. Keys are hexadecimal strings of randomly generated characters, varying in length from 128 to 256 bits. In general, the longer the key, the more secure the encryption implementation.

To discuss TDES, you must first review its predecessor, DES (also referred to as single DES). Starting in 1974, IBM worked with National Institute of Standards and Technology (NIST) and eventually defined Data Encryption Standard (DES), which became an official encryption standard. DES encrypts and decrypts data in 64-bit blocks, using a 64-bit key (although the effective key strength is only 56 bits). It takes a 64-bit block of plain text as input and outputs a 64-bit block of ciphertext. The main algorithm is applied to the plaintext 16 times to produce the ciphertext. With the backing of the NIST, DES became the *de facto* encryption standard and became the prevalent commercially adopted encryption algorithm.

As time progressed, various shortcuts were devised that allowed a DES key to be broken by various brute force attack methods. At the same time, the speed of computing devices increased to the point that the use of a 56-bit key was generally recognized as insufficient to provide adequate protection. By 1997, this situation prompted the NIST to declare that encryption techniques that relied on a DES algorithm were to be considered weak and insecure. Other than in a few situations where systems require only limited security, DES is no longer an acceptable encryption implementation.

Because DES was widely used at the time NIST deprecated its use, an expedient solution was to introduce TDES as an alternative. TDES, which is recognized by the NIST as providing secure enough protection for most purposes, is a construction of applying DES three times in sequence. TDES, with the use of three different keys (56-bit each), has effective key length of 168 bits.

In May 2002, NIST accepted a second encryption algorithm, the Advanced Encryption Standard, known as AES. AES operates on data in 128-bit blocks, and can apply 128-bit, 192-bit, or 256-bit keys in the algorithm. Depending on the size of the key, the input clear text data is processed 10, 12, or 16 times to generate the ciphertext result.

Through the year 2030, both TDES and the Advanced Encryption Standard (AES) will coexist as NIST-approved algorithms, allowing for a gradual transition to AES. To support these emerging encryption technologies, IBM has developed cryptographic hardware on System z that implements current standards, including TDES (168-bit key) and AES (128, 192, and 256-bit keys).

With DB2 for z/OS, the built-in functions do not support AES and only use 128-bit TDES support.

Prior to DB2 9 for z/OS, encryption support implemented through DRDA and DB2 Connect used 56-bit DES encryption. In DB2 9, 128-bit SSL encryption was available. For example, a trusted context connection could have its encryption level altered to HIGH.

InfoSphere Guardium Data Encryption Tool for IMS and DB2 Databases supports both TDES (128-bit and 168-bit) and AES (128-bit, 192-bit, and 256-bit keys). Both the TS1130 encrypting tape drives and the DS8700 encrypted storage use AES implementations. Network encryption as implemented on DB2 10 for z/OS with either SSL or AT-TLS can support either TDES or AES.

B.2 Integrated Cryptographic Service Facility for z/OS

In the z/OS environment, the Integrated Cryptographic Service Facility (ICSF) provides access to cryptographic functions through callable services. The ICSF callable services comply with the IBM CCA cryptographic API and are available for programs written in assembler or high-level languages. CCA supports a hierarchical structure of keys where keys can be encrypted by other keys (key-encrypting keys (KEKs)), the master key being at the top of the hierarchy.

ICSF provides administration facilities for loading the master keys into the coprocessors.

ICSF also provides key repositories in the form of two VSAM data sets, where keys can be kept in key tokens in clear text or encrypted under the coprocessors master keys. The VSAM data sets are the cryptographic key data set (CKDS) and the public key data set (PKDS). The key tokens in the CKDS and the PKDS are given a user- or system-defined label that is used for their retrieval and maintenance.

In Figure B-2, you can see an application program that has issued a CCA cryptographic API call on an System z9. The call is routed to the ICSF started task. The ICSF started task invokes RACF to determine whether the user ID associated with the request is authorized to use the requested cryptographic service and any keys associated with the request. If the user ID has the proper authority, the ICSF started task decides whether it should perform the request using ICSF software or cryptographic hardware.

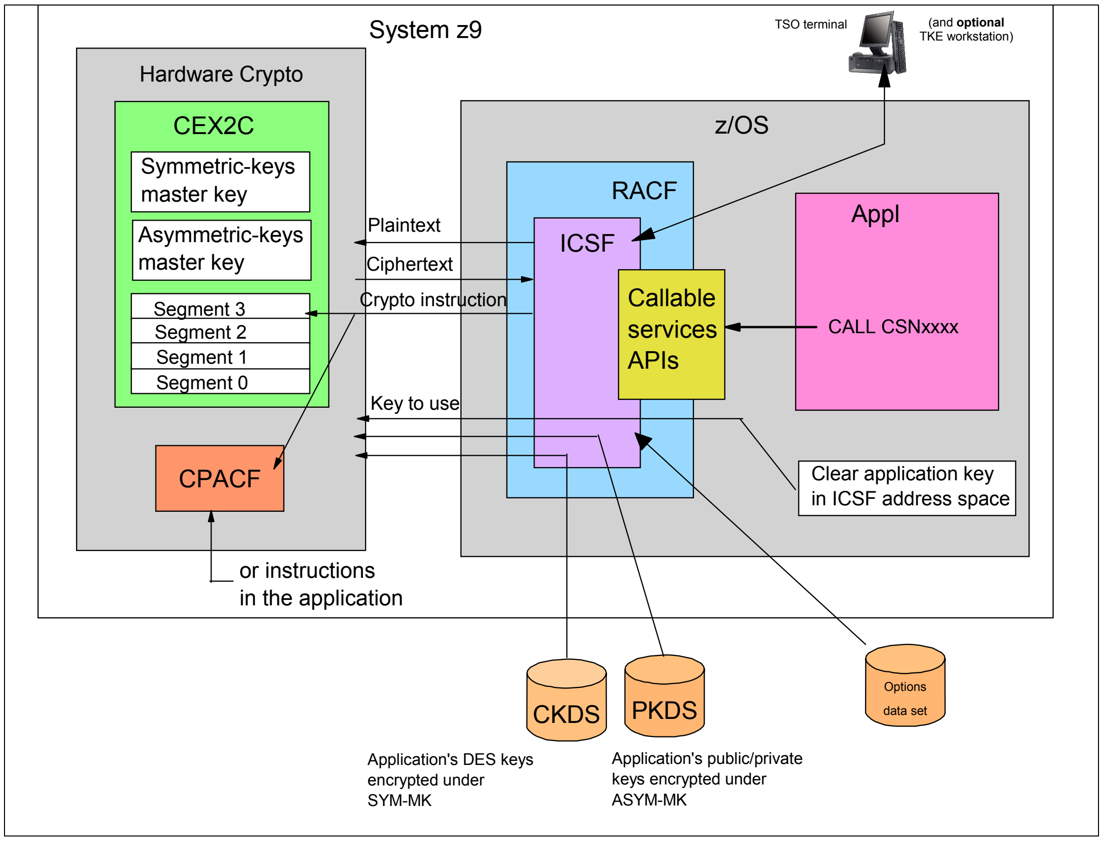

System z9
TSO terminal
(and optional TKE workstation)
Hardware Crypto
CEX2C
Symmetric-keys master key
Asymmetric-keys master key
Segment 3
Segment 2
Segment 1
Segment 0
CPACF
or instructions in the application
z/OS
RACF
ICSF
Callable services APIs
Appl
CALL CSNxxxx
Plaintext
Ciphertext
Crypto instruction
Key to use
Clear application key in ICSF address space
CKDS
PKDS
Options data set
Application's DES keys encrypted under SYM-MK
Application's public/private keys encrypted under ASYM-MK

Figure B-2 Overview of how the hardware and software work together

If ICSF decides to use cryptographic hardware, it gives control to its routines that contain the crypto instructions. The crypto instructions that drive the CEX2C (or CEX3C) are IBM proprietary. IBM does not disclose them.

If ICSF routes the request to the CEX2C and the request is, for example, a request to encrypt data, the ICSF started task provides the CEX2C with the data to be encrypted and the key to be used by the encryption algorithm. Recall that the key is encrypted under a variant of the symmetric key-master key (SYM-MK) stored in the CEX2C.

The interactions between the functional blocks shown in Figure B-2 are as follows:

- ICSF is a z/OS started task that offers cryptographic APIs to applications and drives the requests to the Crypto Express2 Coprocessors (CEX2C).
- The CEX2C is a secure coprocessor in that it contains a master key used to encrypt keys to be kept in the CKDS or PKDS data set. The master key resides in the coprocessor hardware only and is used to decrypt, internally to the coprocessor, the secure keys provided so that they can be used to encrypt or decrypt data.
- ICSF needs also an options data set that contains the ICSF started task startup parameters.

- Installing and maintaining the secret master key is a task that security officers can perform from TSO/E terminals or from an optional Trusted Key Entry (TKE) workstation, the latter for a high security level of the interactions between the security officers and the CEX2C.
- If ICSF has access to more than one secure coprocessor, all coprocessors must have been set with the same master key value.

The CPACF operates only with clear keys. The keys can be stored in ICSF-managed VSAM data sets and pointed to by the application program by using the label under which they are stored. The CKDS is used to store the symmetric keys in their encrypted form and clear keys in their unencrypted form. The PKDS is used to store the asymmetric keys. If the level of ICSF that you are using is HCR7720 or higher, you can also store keys in the CKDS in clear (unencrypted) form.

B.2.1 Clear and secure keys for z/OS

When generating data encrypting keys to be stored in the CKDS or the PKDS, the key officer chooses the form of the key, either a secure or clear key. Choosing a secure key results in a data encrypting key itself being encrypted (or protected) with the DES or AES master key inside the CEX2C/CEX3C hardware feature prior to being stored in the CKDS. In addition, any subsequent access to this key is performed within the CEX2C/CEX3C, and the use of the decrypted data encrypting key as input to the encryption or decryption operation also occurs completely within the CEX2C/CEX3C. This process ensures that at no time is the value of the unencrypted data encrypting key exposed inside of operating system storage. The choice of secure or clear key needs to be communicated to the administrator of InfoSphere Guardium Data Encryption for DB2 and IMS Databases, as the tool can use either type of key. The type of EDITPROC that is prepared depends on the choice of secure or clear key made by the key officer.

Contrasted with secure key processing, clear keys are stored inside the CKDS in clear text format. A clear key encryption request does not run inside the CEX2C/CEX3C; rather, the key is extracted directly from the CKDS as identified by the key label. The data encrypting key is used by the CPACF and clear key encryption and decryption is performed on the CPACF.

The Integrated Cryptographic Service Facility (ICSF) is a component of z/OS. CP Assist for Cryptographic Function (CPACF) is a feature available on IBM System z114, IBM System z196, and IBM System z10, which, while a non-chargeable option, requires that the hardware CE enable its use. When enabled, CPACF adds cryptographic machine instruction support to every general purpose processor, making routine use of cryptography more transparent. CPACF provides support for clear key encryption on the System z hardware. Clear key encryption requests provide much better performance.

Figure B-3 shows a visual representation of clear key encryption.

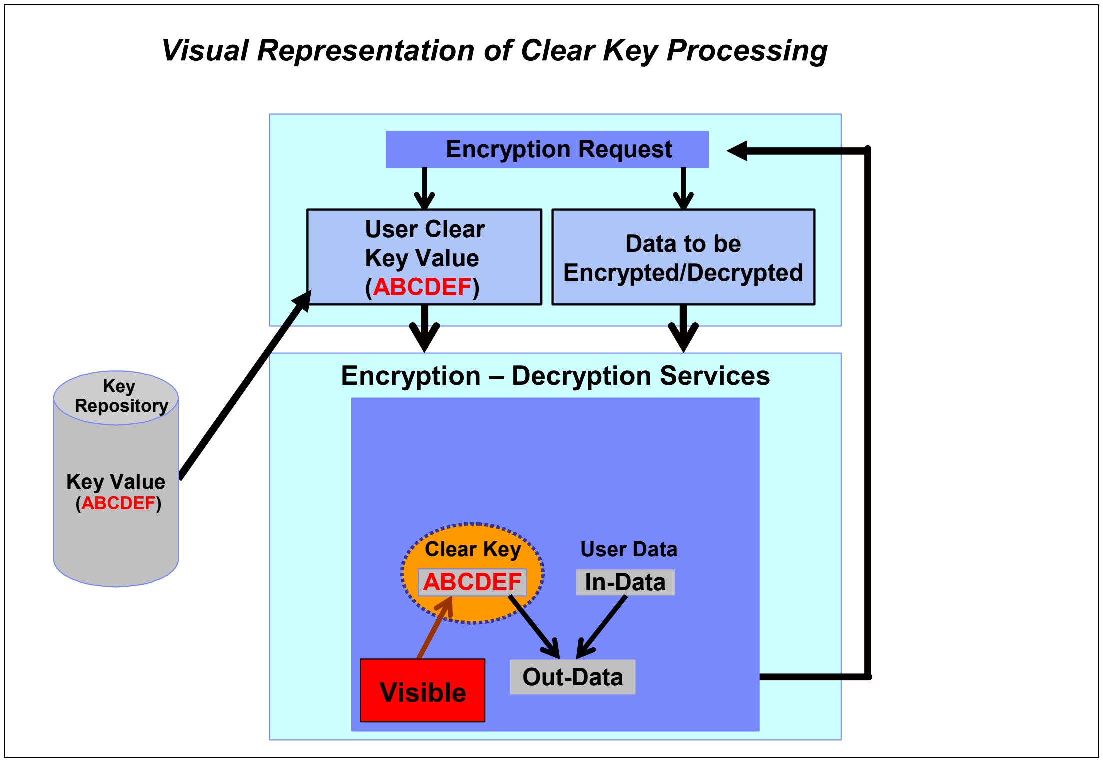

Visual Representation of Clear Key Processing
Encryption Request
User Clear Key Value (ABCDEF)
Data to be Encrypted/Decrypted
Key Repository
Key Value (ABCDEF)
Encryption – Decryption Services
Clear Key
ABCDEF
User Data
In-Data
Out-Data
Visible

Figure B-3 Visual representation of clear key processing

The application requestor passes the key label to ICSF through an ICSF API, which locates the clear key value stored in the key repository, known as the CKDS. Using the extracted key, the application requestor then either performs an encrypt or decrypt request using the appropriate API or through a special machine instruction known as KMC. During clear key encryption processing, the key value is exposed in operating system storage as clear text, as clear keys are not protected under the master key.

Figure B-3 shows the clear key value in storage. Generally the key will only be in the ICSF address space if you are using the ICSF APIs. The only time a clear key would be in the application address space would be if you were using the native CPACF instructions and hardcoded the key into the application, or extracted the clear key value from the key token.

In addition to the CPACF facility, which runs on the general purpose processors, there is an additional hardware element, the CEX2C/CEX3C feature, which can be added for a separate cost. This feature is used to support secure key encryption. All secure key cryptographic work is performed within the hardware boundaries of the CEX2C/CEX3C feature. This features provides a completely isolated environment, and at no point in time is any data or cryptographic key exposed to operating system storage. It is also referred to as tamper resistant. In the event of someone gaining access to the CEX2C/CEX3C hardware, upon attempting to remove the element from the hardware cabinet, the registers containing the master key values are zeroized, thereby protecting the key from inadvertent or intentional exposure or theft.

Figure B-4 shows a representation of secure key encryption processing.

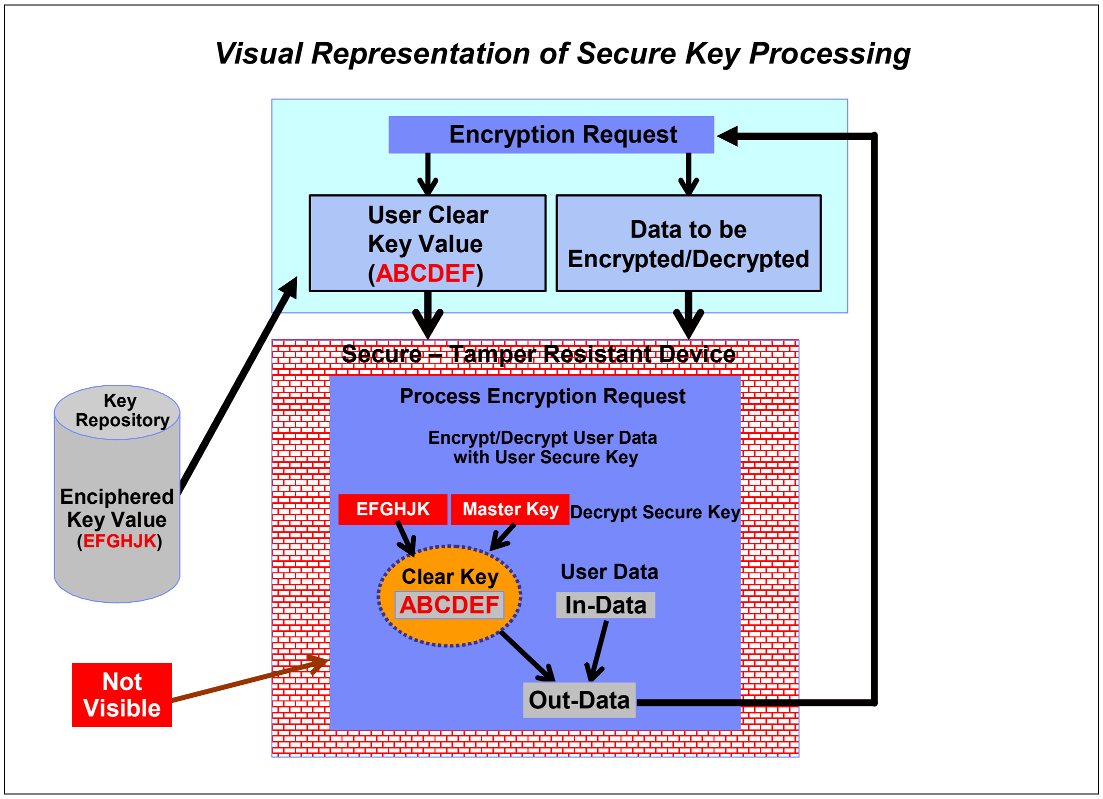

Visual Representation of Secure Key Processing
Encryption Request
User Clear Key Value (ABCDEF)
Data to be Encrypted/Decrypted
Secure – Tamper Resistant Device
Process Encryption Request
Encrypt/Decrypt User Data with User Secure Key
EFGHJK
Master Key
Decrypt Secure Key
Clear Key ABCDEF
User Data
In-Data
Out-Data
Key Repository
Enciphered Key Value (EFGHJK)
Not Visible

Figure B-4 Visual representation of secure key processing

With secure key encryption, the application requestor first provides the keylabel to ICSF, which is then used to locate the corresponding key stored in the CKDS. By definition, secure keys are protected by being encrypted under the AES or DES master key. The application requestor now invokes ICSF passing the keylabel and data. Based on the form of the key, ICSF determines that a secure key is being used, and invokes secure key processing by dispatching control to the CEX2C/CEX3C hardware feature. The data encrypting key will be decrypted by the master key, and the now clear text data key will be used to encrypt or decrypt the user data as appropriate. The result is then passed back to the application requestor.

The following characteristics of this approach are apparent:

- The request to the CEX2C/CEX3C is interrupt driven as the special crypto features are bus attached, so the performance characteristics are similar to that of an I/O request.
- At no time is the data key exposed in operating system storage, as the decryption of the data key under the master key is performed within the confines of the CEX2C/CEX3C.
- Generally, performance of secure key encryption is worse than clear key encryption, especially regarding the elapsed time due to the interrupt driven nature of the interaction with the CEX2C/CEX3C.

Figure B-5 summarizes the different forms of key and encryption types supported by ICSF for z/OS.

	zSeries 900	System z9 and z10		System z10
	CCA Secure Key	Clear Key	CCA Secure Key	CPACF Protected Key
Key Wrapping: Host Storage	CCA Master Key **– Key material is never visible in the clear outside the tamper resistant hardware boundary**	None – **Key material is visible in the clear in system and application storage .**	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**	CPACF Wrapping Key **– Key material is not visible in the clear in *operating system or application storage*.**
Key Wrapping: Key Store	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**	None – **Key material is visible in the clear key store.**	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**	CCA Master Key – **Key material is never visible in the clear outside the tamper resistant hardware boundary**
Key Store	CKDS or *application key file*	CKDS or *application key file*	CKDS or *application key file*	CKDS only
Encryption Engine	CCF	CPACF **or software**	CEX2C	CPACF
Symmetric Encryption Algorithms	DES and TDES	DES, TDES and AES	DES, TDES and AES	DES, TDES and AES
Benefits	***High Performance*** ***High Security***	***High Performance***	***High Security***	***High Performance*** ***High Security***

Figure B-5 System z symmetric key options

InfoSphere Guardium Data Encryption for DB2 and IMS Databases can use clear, secure, and protected key implementations. The DB2 10 for z/OS built-in-function for encryption performs clear key encryption only. System Storage DS8700 and System Storage TS1130 tape encryption perform clear key operations only.

B.2.2 Protected Key support with CEX3C

As described in the previous section, clear key encryption performed on the CPACF generally provides the best performance, particularly for OLTP application workloads. However, when you store clear key values in the CKDS in clear text, in the light of security requirements for protecting key assets, you are using a weak implementation of security. In contrast, secure data encrypting keys are stored in the CKDS as encrypted values protected by the AES or DES master key, which provides a secure repository for key materials. However, cryptographic exploitation of secure keys requires the use of the CEX2C/CEX3C specialty cryptographic hardware features. Given the interrupt based nature of the interaction between the application requestor and the secure hardware, performance characteristics of secure key does not provide adequate application response time, particularly for OLTP or database implementations.

To provide a better implementation, where key asset protection can be provided as well as acceptable performance, a new encryption approach has been introduced. This form of encryption is now known as Protected Key CPACF. It can be viewed as a hybrid implementation combining the performance of clear key CPACF encryption with the key protection of CCAsecure key in CEX2C/CEX3C.

To implement Protected Key CPACF, also known as Message Security Assist (MSA-3), the following prerequisites are required:

- ICSF must be at release level HCR7770 or greater.
- The CEX3C hardware features must be available with the master keys loaded.
- CEX3C requires a System z10, System z196, or System z114 server. z10 EC processors must be on microcode level GA3 or later. z10 BC processors must be on microcode GA2 or greater.

Protected Key CPACF uses secure data encrypting keys that are generated in the usual fashion. In addition, a new key form is introduced, known as the MSA-3 wrapping key. The wrapping key is created when the LPAR where the ICSF has been enabled is activated. It can use an enabled CEX3C feature to use random number generation services. The wrapping key is unique per LPAR, and persists between IPLs. The wrapping key is stored in a protected area of the System z hardware called the Hardware Storage Area (HSA). Figure B-6 gives an overview of the Protected Key CPACF processing.

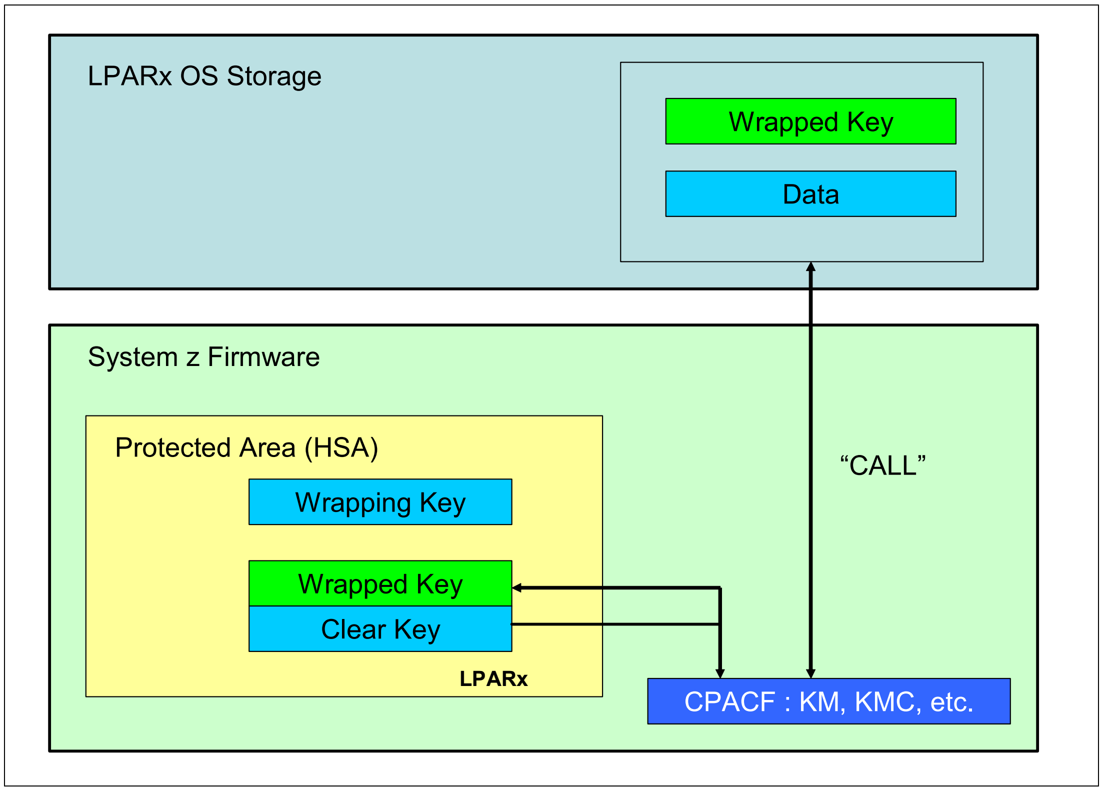


Figure B-6 Protected Key CPACF representation

Protected Key CPACF processing occurs in the following order:

1. An AES or TDES secure key is created by the ICSF key administrator and then is stored in the Cryptographic Key Dataset (CKDS). he key value is a secure key, stored in the CKDS and protected under the AES or DES master key. When stored inside the CKDS, there is an associated key label that is used by requestors to reference the key.

2. The clear key API (CSNBSYE) is invoked and passes the data to be processed along with the key label of the secure key. In prior releases of ICSF, this action would have been rejected by ICSF as an attempt to use a secure key with a clear key request, and resulted in an error. With HCR7770, this action is now recognized as a Protected Key CPACF request and is processed as such.
3. ICSF uses the key label to locate the secure key in the CKDS and pass it to the CEX3C specialty crypto feature.
4. Inside the CEX3C, the secure key is decrypted from under the master key.
5. The retrieved key representation is then wrapped (encrypted) with the wrapping key established for this LPAR.
6. The wrapped key is then passed back to the CPACF.
7. The CPACF then unwraps the wrapped key using the LPAR established wrapping key; this action is performed in the HSA.
8. CPACF then performs clear key processing using the unwrapped key.

Protected Key CPACF provides the ability to achieve a performance that is equivalent to clear key processing in the CPACF, while at the same time ensuring that key assets continue to be protected under the appropriate master key, within the CEX3C specialty hardware. Protected Key CPACF processing is supported for InfoSphere Guardium Data Encryption for DB2 and IMS Databases only.

B.2.3 Key rotation

Key rotation is a concept where, at some pre-determined interval, keys that are used to encrypt data must be changed. Some regulatory compliance initiatives specifically mandate a set period of time, for example, every 12 months. Other times, key rotation needs to be performed whenever a breach or key compromise, whether actual or suspected, has occurred.

There are several different interpretations of this requirement. One is that in an environment where data encrypting keys are stored in a CEX2C/CEX3C protected CKDS, rotation of the DES master key is sufficient. In this scenario, the data encrypting key does not change in value, but the key as stored on the CKDS is encrypted with a different DES master key. This key rotation exercise is performed by the ICSF administrator, can be performed in a nondisruptive manner, and does not require an unload of the data or any type of outage.

The second interpretation of this requirement calls for the data that is protected by the original key to be re-encrypted using a new key value. For this type of rotation to occur, the data must be first unloaded under the old key, then reloaded using the new key.

B.2.4 Protecting ICSF resources with RACF

You can use z/OS Security Server RACF to control which applications can use specific keys and services, which can help you ensure that keys and services are used only by authorized users and jobs. You can also use RACF to audit the use of keys and services. The XCSFKEY class controls who can export a token using the Symmetric Key Export callable service (CSNDSYX). To set up these controls, you create and maintain RACF general resource profiles in the CSFKEYS class, the CSFSERV class, and the XFACILIT class. The CSFKEYS class controls access to cryptographic keys with the key label, the CSFSERV class controls access to ICSF services, and resources in the XFACILIT class define a key store policy that controls the use of key tokens that are stored in the CKDS and PKDS.

To set up profiles in the CSFKEYS general resource class, complete the following steps:

1. Define the appropriate profiles in the CSFKEYS class:

   ```
   RDEFINE CSFKEYS label UACC(NONE) other-optional-operands
   ```

   Label is the label by which the key is defined in the CKDS or PKDS (this is not the transport key label). Note that if an application uses a token instead of a key label, no authorization checking is done on the use of the key.

   – If you have ICSF/MVS Version 1 Release 1 profiles that specify key-type.label, you need to change them to specify only label.

   – As with any RACF profile, if you want to change the profile later, use the RALTER command. To change the access list, use the PERMIT command described in step 2.

   – If you have already started ICSF, you need to refresh the in-storage profiles (see step 4). You can specify other operands, such as auditing (AUDIT operand), on the RDEFINE or RALTER commands.

   – If the RACF security administrator has activated generic profile checking for the CSFKEYS class, you can create generic profiles using the generic characters * and %. These profiles are the same as any RACF general resource class.

2. Give the appropriate users (preferably groups) access to the profiles:

   ```
   PERMIT profile-name CLASS(CSFKEYS) ID(groupid) ACCESS(READ)
   ```

3. Define a RACF profile for the Protected Key. The ICSF extension in the profile must specify SYMCPACFWRAP(YES).

   That tells ICSF that although this key is defined as a secure key, it can be used in the clear, inside the CPACF, as a protected key. If SYMCPACFWRAP(NO) is specified (and that is the default) then the key can only be used as a secure key.

4. When the profiles are ready to be used, ask the RACF security administrator to activate the CSFKEYS class and refresh the in-storage RACF profiles:

   ```
   SETROPTS CLASSACT(CSFKEYS) SETROPTS RACLIST(CSFKEYS) REFRESH
   ```

5. By default, when CSFKEYS is active, all problem state callers of ICSF services using a key resources protected by CSFKEYS are checked by RACF. By default, authorized requestors are not checked. To enable CSFKEYS checking of supervisor state or authorized programs requires the ICSF parameter CHECKAUTH to be set to YES.

6. In a DB2 environment, set CHECKAUTH to NO.

B.3 Tivoli Key Lifecycle Manager

You can use Tivoli Key Lifecycle Manager (TKLM) to manage encryption keys and certificates. TKLM allows you to create, back up, and manage the life cycle of keys and certificates that your enterprise uses, which includes the management of symmetric keys, asymmetric keys, and certificates. Tivoli Key Lifecycle Manager waits for and responds to key generation or key retrieval requests that arrive through TCP/IP communication for a tape library, tape controller, tape subsystem, device drive, tape drive, or DS8000 drive. Tivoli Key Lifecycle Manager provides you with additional functions beyond those offered in the previous IBM key management product (IBM Encryption Key Manager), including:

- Life cycle functions
  - Notification of certificate expiration
  - Automated rotation of certificates
  - Automated rotation of groups of keys

- Usability enhancements: Provides a graphical user interface.
- Initial configuration wizards
  - Migration wizards
  - Provides a command-line interface through WSAdmin
- Integrated backup and restore of Tivoli Key Lifecycle Manager file. There is one button to create and restore a single backup packaged as a JAR file.
- Security policy: Leverages the Security Infrastructure of the IBM System Services Runtime Environment.
- Audit enhancements: Provides audit records in SMF Type 83 sub-type 6 format.

Tivoli Key Lifecycle Manager stores the drive table in DB2, giving the user a more robust interface for managing drives and the keys and certificates that are associated with those drives. With Encryption Key Manager, the previous key management product, which was the only place to determine the key used to encrypt a tape cartridge, and similar audit information, was in the Encryption Key Manager audit log and the Encryption Key Manager `metadata.xml` file. With Tivoli Key Lifecycle Manager, this information is stored in the Tivoli Key Lifecycle Manager DB2 tables, enabling the user to search and query that information with ease.

B.3.1 IBM Java Security keystore

The keystore is defined as part of the Java Cryptography Extension (JCE) and is an element of the IBM Java Security components, which are, in turn, part of the Java Runtime Environment. A keystore holds the certificates and keys (or pointers to the certificates and keys) used by Tivoli Key Lifecycle Manager to perform cryptographic operations. A keystore can be either hardware-based or software-based. Tivoli Key Lifecycle Manager supports several types of Java keystores, offering a variety of operational characteristics to meet your needs.

B.3.2 Tivoli Key Lifecycle Manager on distributed systems

Tivoli Key Lifecycle Manager on distributed systems supports the JCEKS keystore. This keystore supports both symmetric keys and asymmetric keys. Symmetric keys are used for LTO-4[1] encryption drives, while asymmetric keys are used for the TS1100 family of tape drives and the DS8000 drives.

B.3.3 Cryptographic services

Tivoli Key Lifecycle Manager uses the IBM Java Security components for its cryptographic capabilities. Tivoli Key Lifecycle Manager does not provide cryptographic capabilities and therefore does not require, or is allowed to obtain, FIPS 140-2 certification. However, Tivoli Key Lifecycle Manager takes advantage of the cryptographic capabilities of the IBM Java Virtual Machine in the IBM Java Cryptographic Extension component and allows the selection and use of the IBMJCEFIPS cryptographic provider, which has a FIPS 140-2 level 1 certification. By setting the FIPS configuration parameter to ON in the Configuration Properties file, either through text editing or using the Tivoli Key Lifecycle Manager CLI, you can make Tivoli Key Lifecycle Manager use the IBMJCEFIPS provider for all cryptographic functions.

[1] Linear Tape-Open (LTO) is a magnetic tape data storage technology developed as an open standard of magnetic tape formats. LTO-4 is the fourth generation of drives, which introduced drive level encryption feature using 256-bit AES-GCM: IBM Ultrium Generation 4.

For more information about the IBMJCEFIPS provider, its selection and use, go to the following address:

http://www.ibm.com/developerworks/java/jdk/security/50/FIPShowto.html

B.3.4 Key exchange

Tivoli Key Lifecycle Manager acts as a process awaiting key generation or key retrieval requests sent to it through a TCP/IP communication path between Tivoli Key Lifecycle Manager and the tape library, tape controller, tape subsystem, device driver, tape drive, or DS8000 drive. When a drive writes encrypted data, it first requests an encryption key from Tivoli Key Lifecycle Manager. The tasks that the Tivoli Key Lifecycle Manager performs upon receipt of the request are different for the asymmetric keys used by the TS1100 family of tape drives and the DS8000 drives, and symmetric keys used by the TS1040 tape drive[2].

B.3.5 Asymmetric and symmetric keys

Tivoli Key Lifecycle Manager requests an Advanced Encryption Standard (AES) key from the cryptographic services and serves it to the drives in one of the following forms:

- Encrypted or wrapped, using Rivest-Shamir-Adleman (RSA) key pairs. This form is used for the TS1100 family of tape drives and the DS8000 drives.
- Separately wrapped for secure transfer to the tape drive, where it is unwrapped upon arrival and the key inside is used to encrypt the data being written to tape. This form is used for the TS1040 tape drives.

Additionally, the libraries now support SSL-encrypted connections between the Tivoli Key Lifecycle Manager and library for key exchanges. When SSL is not used for key exchange, the key material is encrypted in another fashion. The transport of the keys is always secure across the TCP/IP connection.

B.3.6 TS1100 family of tape drives and DS8000

When an encrypted tape cartridge is read by a TS1100 tape drive, the protected AES key on the tape is sent to Tivoli Key Lifecycle Manager, where the wrapped AES key is unwrapped. The AES key is then wrapped with a different key for secure transfer back to the tape drive, where it is unwrapped and used to decrypt the data stored on the tape. Tivoli Key Lifecycle Manager also allows protected AES keys to be re-wrapped, or re-keyed, using different RSA keys from the original keys that were used when the tape was written. Re-keying is useful when an unexpected need arises to export volumes to an organization whose public keys were not included; it eliminates the need to rewrite the entire tape and enables a tape cartridge's data key to be re-encrypted with the organization's public key. Re-keying of the DS8000 is currently not available and would require a complete re-initialization of the drive.

B.3.7 LTO Ultrium 4 tape drives

The Tivoli Key Lifecycle Manager fetches an existing AES key from a keystore and wraps it for secure transfer to the tape drive, where it is unwrapped upon arrival and used to encrypt the data being written to tape. When an encrypted tape is read by an LTO Ultrium 4 tape drive, the Tivoli Key Lifecycle Manager fetches the required key from the keystore, based on the information in the Key ID on the tape, and serves it to the tape drive wrapped for secure transfer.

[2] TS1040 Tape Drive provides an Ultrium 4 Tape Drive for the TS3500 Tape Library.

B.3.8 System-managed encryption for System z

On z/OS, policies specifying when to use encryption are set up in DFSMS. You can also use additional software products, such as ICSF and RACF. Key generation and management is performed by the Tivoli Key Lifecycle Manager, running on the host or externally on another host. Policy controls and keys pass through the data path between the system layer and the encrypting tape drives. Encryption is transparent to the applications.

For TS1120 tape drives that are connected to an IBM Virtualization Engine TS7700,encryption key labels are assigned using the Maintenance Interface on a per-storage-pool basis. DFSMS storage constructs are used by z/OS to control the use of storage pools for logical volumes, resulting in an indirect form of encryption policy management. For more information, refer to the white paper *IBM Virtualization Engine TS7700 Series Encryption Overview*, which is available at the following address:

`http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504`

For details about setting up system-managed encryption on the TS1120 tape drive in a System z platform environment, refer to *z/OS DFSMS Software Support for IBM System Storage TS1120 Tape Drive (3592)*, SC26-7514.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *ABCs of z/OS System Programming Volume 2*, SG24-6982
- *DB2 10 for z/OS Performance Topics*, SG24-7942
- *DB2 10 for z/OS Technical Overview*, SG24-7892
- *IBM eServer zSeries 990 (z990) Cryptography Implementation*, SG24-7070
- *IBM System Storage Tape Encryption Solutions*, SG24-7320
- *IBM System z10 Enterprise Class Technical Guide*, SG24-7516
- *IBM Virtualization Engine TS7500: Planning, Implementation, and Usage Guide*, SG24-7520
- *Securing DB2 and Implementing MLS on z/OS*, SG24-6480
- *Security on the IBM Mainframe*, SG24-7803
- *DB2 9 for z/OS: Configuring SSL for Secure Client-Server Communications*, REDP-4630

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *DB2 10 for z/OS Administration Guide,* SC19-2968
- *DB2 10 for z/OS Application Programming and SQL Guide,* SC19-2969
- *DB2 10 for z/OS Application Programming Guide and Reference for Java,* SC19-2970
- *DB2 10 for z/OS Codes,* GC19-2971
- *DB2 10 for z/OS Command Reference,* SC19-2972
- *DB2 10 for z/OS Installation and Migration Guide,* GC19-2974
- *DB2 10 for z/OS Messages,* GC19-2979
- *DB2 10 for z/OS RACF Access Control Module Guide,* SC19-2982
- *DB2 10 for z/OS SQL Reference,* SC19-2983
- *DB2 10 for z/OS Utility Guide and Reference,* SC19-2984

- *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Version 5.1 Configuration and Customization,* GH12-6928
- *IBM Tivoli OMEGAMON XE for DB2 Performance Monitor for z/OS Version 5.1 Report Reference,* SH12-6921
- *z/OS V1R11 MVS Diagnosis Tools and Service Aids*, GA22-7589
- *z/OS V1R11 Security Server RACF Command Language Reference*, SA22-8776
- *z/OS V1R12.0 Security Server RACF Command Language Reference*, SA22-7687
- *z/OS V1R12 Communications Server: IP Diagnosis Guide*, GC31-8782

Online resources

These websites are also relevant as further information sources:

- *Best Practices IBM Data Server Security*:

  http://public.dhe.ibm.com/software/dw/dm/db2/bestpractices/DB2BP_Security_1010I.pdf
- DB2 10 for z/OS:

  http://www.ibm.com/software/data/db2/zos/
- DB2 Information Center:

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp
- Encryption Facility for z/OS:

  http://www.ibm.com/systems/z/os/zos/encryption_facility/
- IBM InfoSphere Guardium Database Security

  http://www.ibm.com/software/data/guardium/
- *A Matter of Time: Temporal Data Management in DB2 for z/OS*:

  ftp://aix.software.ibm.com/software/data/sw-library/db2/zos/A_Matter_of_Time_-_DB2_zOS_Temporal_Tables_-_White_Paper_v1.4.1.pdf

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Abbreviations and acronyms

AC	autonomic computing
ACS	automatic class selection
AIX	IBM Advanced Interactive eXecutive
APAR	authorized program analysis report
API	application programming interface
AR	application requester
ARM	automatic restart manager
AS	application server
ASCII	American National Standard Code for Information Interchange
BCDS	DFSMShsm backup control data set
BCRS	business continuity recovery services
BI	Business Intelligence
BLOB	binary large objects
BPA	buffer pool analysis
BRF	basic row format
BSDS	boot strap data set
CBU	Capacity BackUp
CCA	channel connection address
CCA	client configuration assistant
CCF	Crytptographic Coprocessor Facility
CCP	collect CPU parallel
CCSID	coded character set identifier
CD	compact disk
CDW	central data warehouse
CF	coupling facility
CFCC	coupling facility control code
CFRM	coupling facility resource management
CICS	Customer Information Control System
CLI	call level interface
CLOB	character large object
CLP	command-line processor
CM	conversion mode
CMOS	complementary metal oxide semiconductor
CP	central processor
CPU	central processing unit
CRCR	conditional restart control record
CRD	collect report data
CRM	customer relationship management
CRUD	create, retrieve, update or delete
CSA	common storage area
CSF	Integrated Cryptographic Service Facility
CSWL	CONCENTRATE STATEMENTS WITH LITERALS
CTE	common table expression
CTT	created temporary table
CUoD	Capacity Upgrade on Demand
DAC	discretionary access control
DASD	direct access storage device
DB	database
DB2	IBM Database 2
DBA	database administrator
DBAT	database access thread
DBCLOB	double-byte character large object
DBCS	double-byte character set
DBD	database descriptor
DBID	database identifier
DBM1	database master address space
DBRM	Database request module
DCL	data control language
DDCS	distributed database connection services
DDF	distributed data facility
DDL	data definition language
DES	Data Encryption Standard
DLL	dynamic load library manipulation language
DML	data manipulation language
DNS	domain name server
DPSI	data partitioning secondary index
DRDA	Distributed Relational Data Architecture
DSC	dynamic statement cache, local or global
DSNZPARMs	DB2 system configuration parameters

DSS	decision support systems
DTT	declared temporary tables
DWDM	dense wavelength division multiplexer
DWT	deferred write threshold
EA	extended addressability
EAI	enterprise application integration
EAS	Enterprise Application Solution
EBCDIC	extended binary coded decimal interchange code
ECS	enhanced catalog sharing
ECSA	extended common storage area
EDM	environmental descriptor manager
EJB	Enterprise JavaBean
ELB	extended long busy
ENFM	enable-new-function mode
ERP	enterprise resource planning
ERP	error recovery procedure
ESA	Enterprise Systems Architecture
ESP	Enterprise Solution Package
ESS	IBM Enterprise Storage Server®
ETR	external throughput rate
EWLC	Entry Workload License Charges
EWLM	IBM Enterprise Workload Manager™
FIFO	first in first out
FLA	fast log apply
FTD	functional track directory
FTP	File Transfer Program
GB	gigabyte (1,073,741,824 bytes)
GBP	group buffer pool
IBM GDPS®	IBM Geographically Dispersed Parallel Sysplex™
GLBA	Gramm-Leach-Bliley Act of 1999
GRS	global resource serialization
GUI	graphical user interface
HALDB	High Availability Large Databases
HPJ	high performance Java
HTTP	Hypertext Transfer Protocol
HW	hardware
I/O	input/output
IBM	International Business Machines Corporation
ICF	internal coupling facility
ICF	integrated catalog facility
ICMF	integrated coupling migration facility
ICSF	Integrated Cryptographic Service Facility
IDE	integrated development environments
IFCID	instrumentation facility component identifier
IFI	Instrumentation Facility Interface
IFL	Integrated Facility for Linux
IMS	Information Management System
IORP	I/O Request Priority
IPL	initial program load
IPLA	IBM Program Licence Agreement
IRD	Intelligent Resource Director
IRLM	internal resource lock manager
IRWW	IBM Relational Warehouse Workload
ISPF	interactive system productivity facility
ISV	independent software vendor
IT	information technology
ITR	internal throughput rate
ITSO	International Technical Support Organization
IU	information unit
IVP	installation verification process
J2EE	Java 2 Enterprise Edition
JDBC	Java Database Connectivity
JFS	journaled file systems
JNDI	Java Naming and Directory Interface
JTA	Java Transaction API
JTS	Java Transaction Service
JVM	Java Virtual Machine
KB	kilobyte (1,024 bytes)
LCU	Logical Control Unit
LDAP	Lightweight Directory Access Protocol
LOB	large object
LPAR	logical partition
LPL	logical page list
LRECL	logical record length
LRSN	log record sequence number
LRU	least recently used
LSS	logical subsystem
LUW	logical unit of work

LVM	logical volume manager
MAC	mandatory access control
MB	megabyte (1,048,576 bytes)
MBps	megabytes per second
MLS	multi-level security
MQT	materialized query table
MTBF	mean time between failures
MVS	Multiple Virtual Storage
NALC	New Application License Charge
NFM	new-function mode
NFS	Network File System
NPI	non-partitioning index
NPSI	nonpartitioned secondary index
NVS	non volatile storage
ODB	object descriptor in DBD
ODBC	Open Database Connectivity
ODS	Operational Data Store
OLE	Object Link Embedded
OLTP	online transaction processing
OP	online performance
OSC	optimizer service center
PAV	parallel access volume
PCICA	Peripheral Component Interface Cryptographic Accelerator
PCICC	PCI Cryptographic Coprocessor
PDS	partitioned data set
PIB	parallel index build
PLSD	page level sequential detection
PPRC	Peer-to-Peer Remote Copy
IBM PR/SM™	Processor Resource/System Manager
PSID	pageset identifier
PSP	preventive service planning
PTF	program temporary fix
PUNC	possibly uncommitted
PWH	Performance Warehouse
QA	Quality Assurance
QMF	IBM Query Management Facility™
QoS	Quality of Service
QPP	Quality Partnership Program
RACF	Resource Access Control Facility
RAS	reliability, availability and serviceability
RBA	relative byte address
RBLP	recovery base log point
RDBMS	relational database management system
RDS	relational data system
RECFM	record format
RI	Referential Integrity
RID	record identifier
ROI	return on investment
RPO	recovery point objective
RR	repeatable read
RRF	reordered row format
RRS	resource recovery services
RRSAF	resource recovery services attach facility
RS	read stability
RTO	recovery time objective
RTS	Real-time statistics
SAN	storage area network
SBCS	store single byte character set
SCT	skeleton cursor table
SDM	System Data Mover
SDP	Software Development Platform
SLA	service level agreement
SMIT	System Management Interface Tool
SOA	service-oriented architecture
SPL	selective partition locking
SPT	skeleton plan table
SQL	Structured Query Language
SQLJ	Structured Query Language for Java
SRM	IBM Service Request Manager®
SSL	Secure Sockets Layer
SU	Service Unit
TCO	total cost of ownership
TPF	Transaction Processing Facility
UA	Unit Addresses
UCB	Unit Control Block
UDB	Universal Database
UDF	user-defined functions
UDT	user-defined data types
UOW	unit of work
UR	uncommitted read
UR	unit of recovery
UTS	universal table space
VCF	virtual coupling facility
VIPA	Virtual IP Addressing

VLDB	very large database
VM	virtual machine
VSE	Virtual Storage Extended

Index

Security Functions of IBM DB2 10 for z/OS

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Security Functions of IBM DB2 10 for z/OS

Implement separation of duties

Audit application and system activity

Protect from intrusions and misplacements

IBM DB2 9 and 10 for z/OS have added functions in the areas of security, regulatory compliance, and audit capability that provide solutions for the most compelling requirements.

DB2 10 enhances the DB2 9 role-based security with additional administrative and other finer-grained authorities and privileges. This authority granularity helps separate administration and data access that provide only the minimum appropriate authority.

The authority profiles provide better separation of duties while limiting or eliminating blanket authority over all aspects of a table and its data. In addition, DB2 10 provides a set of criteria for auditing for the possible abuse and overlapping of authorities within a system.

In DB2 10, improvements to security and regulatory compliance focus on data retention and protecting sensitive data from privileged users and administrators. Improvements also help to separate security administration from database administration.

DB2 10 also lets administrators enable security on a particular column or particular row in the database complementing the privilege model.

This IBM Redbooks publication provides a detailed description of DB2 10 security functions from the implementation and usage point of view. It is intended to be used by database, audit, and security administrators.

SG24-7959-00 ISBN 0738436038