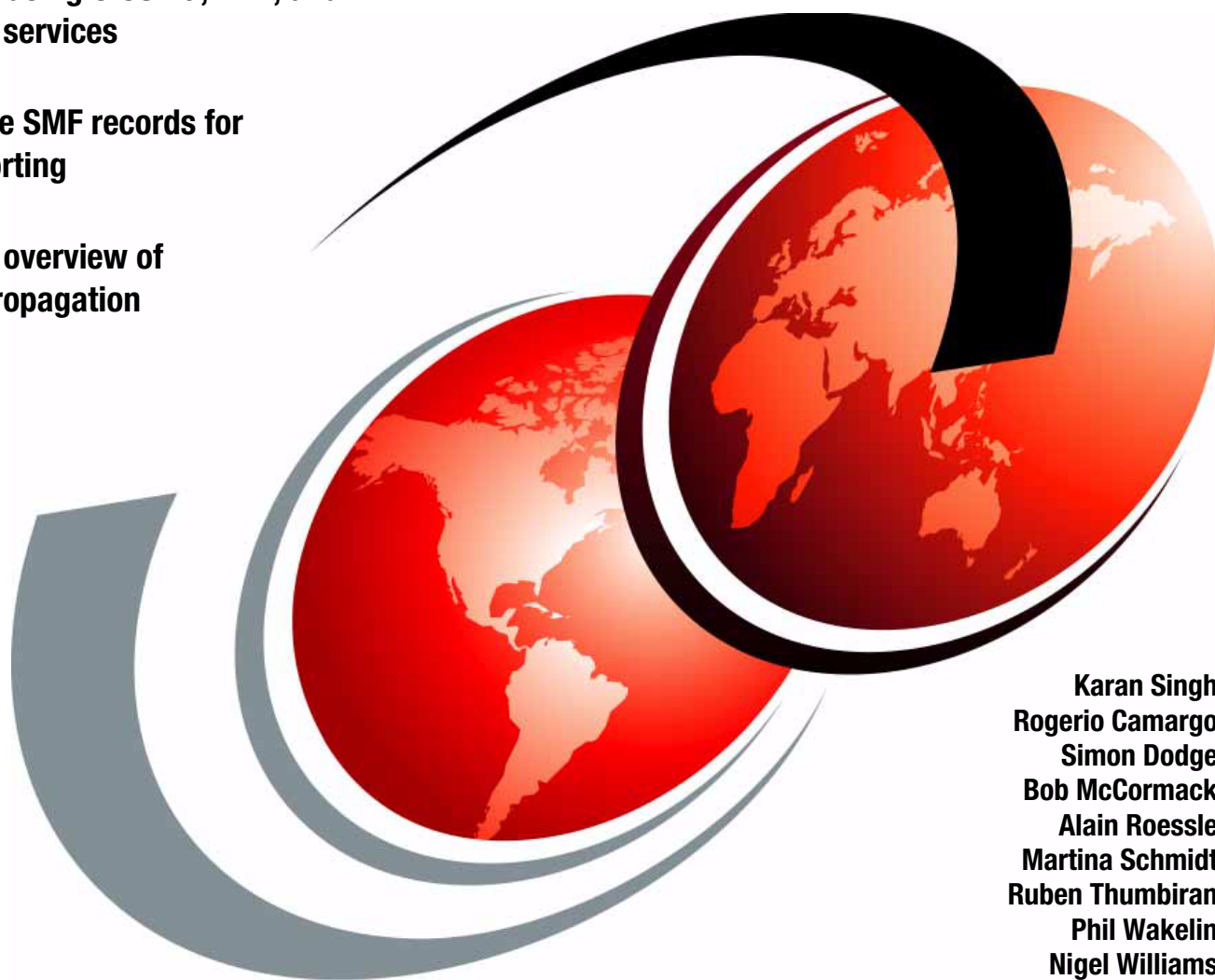


z/OS Identity Propagation

Scenarios using CICS TG, DB2, and CICS Web services

How to use SMF records for audit reporting

Technical overview of identity propagation



Karan Singh
Rogerio Camargo
Simon Dodge
Bob McCormack
Alain Roessle
Martina Schmidt
Ruben Thumbiran
Phil Wakelin
Nigel Williams

Redbooks



International Technical Support Organization

z/OS Identity Propagation

September 2011

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

First Edition (September 2011)

This edition applies to Version 1, Release 11, Modification 0 of z/OS (product number 5694-A01).

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
The team who wrote this book	ix
Now you can become a published author, too!	x
Comments welcome	xi
Stay connected to IBM Redbooks	xi
Chapter 1. Introduction	1
1.1 What z/OS Identity Propagation is	2
1.2 Why identity propagation is important	3
1.2.1 Complete end-to-end audit trail	4
1.2.2 Management of z/OS identity	4
1.2.3 Elimination of the perception of z/OS identity as a weak link	5
1.2.4 Making the distributed identity available to an application program	5
Chapter 2. RACF and z/OS Identity Propagation	7
2.1 Conceptual overview	8
2.1.1 Authentication, distinguished name, and registry	9
2.1.2 Distributed Identity Data Structure (IDID)	11
2.1.3 SAF interfaces	12
2.1.4 R_cacheserv and caching	13
2.1.5 Recreating a security environment or cleaning up a cache	13
2.2 Internal logic within SAF call	13
2.3 Impact on distributed identity data cached in RACF by z/OS subsystems	16
2.4 RACF database unload	17
2.5 RACF messages	19
2.6 RACF templates	19
2.7 RACF remove ID utility	20
2.8 Supplied samples	21
2.8.1 IKJTSO00 in RACPARM	21
2.8.2 Sample DB2 members	22
2.8.3 XML schema document	23
Chapter 3. z/OS Identity Propagation exploiters	25
3.1 CICS Transaction Gateway	26
3.1.1 JCA and security	26
3.1.2 CICS TG for z/OS security options	27
3.1.3 z/OS Identity Propagation support with CICS TG	28
3.1.4 CICS resources used for configuring identity propagation with CICS TG	29
3.2 CICS Web services	29
3.2.1 CICS resources used for configuring web services	30
3.2.2 Securing CICS Web services	31
3.2.3 WebSphere DataPower	31
3.2.4 Identity propagation with CICS Web services	32
3.3 DB2 10 for z/OS	34
3.3.1 DB2 10 for z/OS Trusted Context	35
3.3.2 RACMAP	36

3.3.3	DB2 role in RACF	36
3.3.4	SMF reporting	37
3.3.5	Distinguished name not matching	38
Chapter 4.	RACMAP function	39
4.1	Distributed identity filters	40
4.1.1	What a distributed identify filter is	40
4.2	RACMAP command overview	40
4.3	Authorization required to use the RACMAP command	40
4.4	RACMAP command usage and invocation	41
4.5	Activating the RACMAP updates	43
4.6	RACMAP profiles in the IDIDMAP class	43
4.7	Updating a distributed identity filter	44
4.7.1	Steps for updating a distributed identity filter	44
4.8	User profiles and RACMAP command	45
4.8.1	Deleting a RACF user ID associated with identity filters	45
4.8.2	Performance consideration when deleting a distributed identity filter	45
4.8.3	RACF remove ID utility IRRRID00 update	46
4.9	Default RACMAP filter protection	46
4.10	RRSF consideration for RACMAP use	46
4.11	Changes required to PARMLIB to support identity filter	46
4.12	New RACMAP messages	47
Chapter 5.	Filter management	49
5.1	How RACF matches the filter value	50
5.2	Details about searching for a filter that matches a user's DN	50
5.2.1	One-to-one match	50
5.2.2	Many-to-one match	51
5.2.3	Summary details about searching for a filter that matches a user's DN	52
5.3	Examples	52
Chapter 6.	Using SMF audit information to report on z/OS Identity Propagation	55
6.1	Actions within RACF	56
6.1.1	An RACF event relating to issuing a RACMAP command	56
6.1.2	RACF events when calling RACF to verify a distributed identity	56
6.1.3	Settings within RACF to ensure identity propagation is captured	57
6.2	SMF changes to support distributed identities	57
6.2.1	SMF records	58
6.2.2	SMF unload utility	59
6.3	Reporting from SMF data	60
6.3.1	SMF unload utility	60
6.3.2	SMF UNLOAD produces XML data	61
6.3.3	Reporting on SMF audit information from DB2	63
6.3.4	Using ICETOOL from DFSORT	64
6.3.5	Using IBM Security zSecure Audit for RACF	68
Chapter 7.	Internal z/OS data structures impacted by identity propagation	77
7.1	SAF interfaces	78
7.1.1	RACF communication vector table	78
7.1.2	RACROUTE	78
7.1.3	InitACEE	79
7.1.4	R_cacheserv	80
7.1.5	R_usermap	81
7.2	SAF data areas	81

7.2.1	Accessor Environment Element	81
7.2.2	ICRX: Extended Identity Context Reference	82
7.2.3	IDID: Distributed Identity Data	83
7.2.4	ENF2: RACF ENF Event Code 71	83
Chapter 8.	Identity propagation with CICS and CICS Transaction Gateway	85
8.1	Architectural overview	86
8.2	Configuring identity propagation on CICS Transaction Gateway	87
8.3	Configuring identity propagation on CICS Transaction Server	89
8.4	Configuring identity propagation on WebSphere Application Server	90
8.4.1	Configuring standalone LDAP registry	91
8.4.2	Deploying the CICS ECI resource adapter	94
8.4.3	Creating a J2C Connection Factory	96
8.4.4	Deploying the ECIDateTime application	98
8.4.5	Installing the identity propagation login module	100
8.4.6	Running the ECIDateTime application	102
8.5	Configuring identity propagation on z/OS	105
8.6	Configuring identity propagation on RACF	106
8.7	Testing the scenario	106
8.7.1	Results for No Distributed Identity Passed	106
8.7.2	Results for No mapping found	107
8.7.3	Results for one-to-one mapping	108
8.7.4	Results for many-to-one mapping	109
8.7.5	Results of DPL to second CICS showing INQUIRE ASSOCIATION	110
8.7.6	Results after RACF mappings have changed	112
Chapter 9.	Identity propagation with DB2 for z/OS	113
9.1	JAVA application test scenario	114
9.2	RACMAP command	115
9.3	Creating DB2 trusted context	116
9.4	Creating the DB2 role	116
9.5	RACF/DB2 exit (optional)	116
9.6	Executing the sample Java application	117
9.7	RACF audit trace	119
9.8	DB2 Audit trail	121
Chapter 10.	Identity propagation using CICS Web services	123
10.1	Scenario overview	124
10.2	Architectural overview	126
10.3	Preparation	127
10.3.1	Software versions	127
10.3.2	IP addresses and ports	127
10.3.3	CICS resource definition checklist	127
10.3.4	User IDs	128
10.3.5	Keystore and certificates	128
10.4	Configuring RDz	129
10.5	Configuring WebSphere DataPower	132
10.6	Configuring CICS	148
10.6.1	SIT parameters	148
10.6.2	TCPIPSERVICE	148
10.6.3	URIMAP	149
10.6.4	PIPELINE	150
10.6.5	WEBSERVICE	151
10.6.6	MRO connection	152

10.7	Configuring RACF	153
10.7.1	Identity mapping rules	153
10.7.2	Authorizing the service requester	153
10.8	Testing the scenario	155
10.8.1	Successful many-to-one identity mapping	155
10.8.2	Failure scenarios	159
	Related publications	161
	IBM Redbooks	161
	Other publications	161
	Help from IBM	161
	Index	163

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS Explorer™	IBM®	System z®
CICSplex®	MVS™	Tivoli®
CICS®	Parallel Sysplex®	VTAM®
DataPower device®	RACF®	WebSphere®
DataPower®	Rational®	z/OS®
DB2 Connect™	RDN®	z/VM®
DB2®	Redbooks®	zSeries®
DRDA®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication explores various implementations of z/OS® Identity Propagation where the distributed identity of an end user is passed to z/OS and used to map to a RACF® user ID, and any related events in the audit trail from RACF show both RACF and distributed identities.

This book describes the concept of identity propagation and how it can address the end-to-end accountability issue of many customers. It describes, at a high level, what identity propagation is, and why it is important to us. It shows a conceptual view of the key elements necessary to accomplish this.

This book provides details on the RACMAP function, filter management and how to use the SMF records to provide an audit trail. In depth coverage is provided about the internal implementation of identity propagation, such as providing information about available callable services.

This book examines the current exploiters of z/OS Identity Propagation and provide several detailed examples covering CICS® with CICS Transaction Gateway, DB2®, and CICS Web services with Datapower.

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, Poughkeepsie Center.

Karan Singh is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of z/OS.

Rogério Camargo is a z/OS System Support and RACF Security Specialist in IBM Global Services in Brazil. He has 10+ years of experience in the IT Security Mainframe platform. His areas of expertise include security assessment, penetration test, security process, compliance, and health checking. He has performed several security projects, mainly in large bank institutions in Mexico, Brazil, and Greece. He teaches z/OS Security Server classes in Brazil, and he was also one of the authors of the IBM Redbooks publication *z/OS Version 1 Release 8 RACF Implementation*, SG24-7248.

Simon Dodge is a zSeries® Security Architect/Engineer working for WellsFargo Bank in the USA. He has 20 years of experience working with RACF and 12 years working with CICS for financial institutions, both as an Application Developer and a Systems Programmer. He has a degree in electrical engineering (applied science) from the University of Toronto. He speaks regularly at RACF user groups and SHARE. One of his interests is exploiting the functionality of the zSecure product suite to satisfy non-trivial reporting needs, both for internal and regulatory requirements. User identities have been of interest to him ever since he migrated CICS/MVS™ systems to use RACF back in the late 1980s.

Bob McCormack is an Advisory Software Engineer at the IBM Australian Development Laboratory and has worked on many z/OS and z/VM® products in a wide variety of capacities. He has a Bachelor of Applied Science degree from the University of Technology,

Sydney and is an IBM Certified IT Specialist. He joined IBM in 2007 after many years with IBM business partners.

Alain Roessle is a Certified IT Specialist working in the IBM Design Centre, Montpellier, France. Before joining IBM in 2000, he worked for a large distribution company for 20 years. His areas of expertise include WebSphere®, CICS, DB2, and Parallel Sysplex®.

Martina Schmidt is a Senior Client Technical Specialist in System z® Software Technical Sales in Germany. She has five years of experience in mainframes and holds a Bachelor in Applied Computer Science degree from the University of Cooperative Education in Stuttgart. Martina's areas of expertise include z/OS Security, IBM Security zSecure Product Suite, and RACF.

Ruben Thumbiran has over 20 years of experience on IBM Mainframes, working as a MVS Systems Programmer, a CICS Systems Programmer, Capacity and Planning, and DB2 on z/OS. For the past six years he has worked in IBM Global Services in South Africa as an IT Specialist, DB2 Database Administration. Ruben is a Certified DB2 for z/OS V8 Administrator.

Phil Wakelin works for IBM UK in Hursley and is a member of the CICS Strategy and Planning Team. He has worked with many CICS technologies over the last 20 years. Currently, he is responsible for planning new functionality in the areas of CICS interconnectivity, CICS Java support, and future releases of the CICS Transaction Gateway. He has authored many whitepapers, SupportPacs, and IBM Redbooks.

Nigel Williams is a Certified IT Specialist working in the IBM Design Centre, Montpellier, France. He specializes in enterprise application integration, security, and service-oriented architectures. He is the author of many papers and IBM Redbooks publications, and he speaks frequently on CICS and application integration topics.

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

Richard Conway, Robert Haimowitz
International Technical Support Organization, Poughkeepsie Center

George Markouizos, Russ Hardgrove
IBM RACF

James Pickel, Derek Tempongko
IBM Software Group

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
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>



Introduction

This publication explores various implementations of z/OS Identity Propagation where the distributed identity of an end user is passed to z/OS and used to map to a RACF user ID, and any related events in the audit trail from RACF show both RACF and distributed identities.

This chapter describes the concept of identity propagation and how it can address the end-to-end accountability issue of many customers. This chapter describes, at a high level, what identity propagation is and why it is important. This chapter shows a conceptual view of the key elements necessary to accomplish this.

Subsequent chapters explore in detail exactly what level of software support is available. It is expected that there will be a gradual exploitation of z/OS Identity Propagation by various subsystems.

1.1 What z/OS Identity Propagation is

In today's heterogeneous computing environment, often transactions are initiated outside of the mainframe environment but eventually run within a z/OS subsystem such as CICS or DB2. In a typical case (Figure 1-1), a user initiates a transaction using her distributed identity. When this work enters z/OS, the user-distributed identity is lost because in the z/OS environment only a z/OS security server RACF user ID is relevant. Commonly, transactions running in a z/OS subsystem (for example, CICS) execute under a single z/OS RACF user ID, although the users initiating the transactions might be different. There is no association between the distributed identity of the user who starts the transaction and the z/OS Security Server RACF user ID under which the transaction runs.

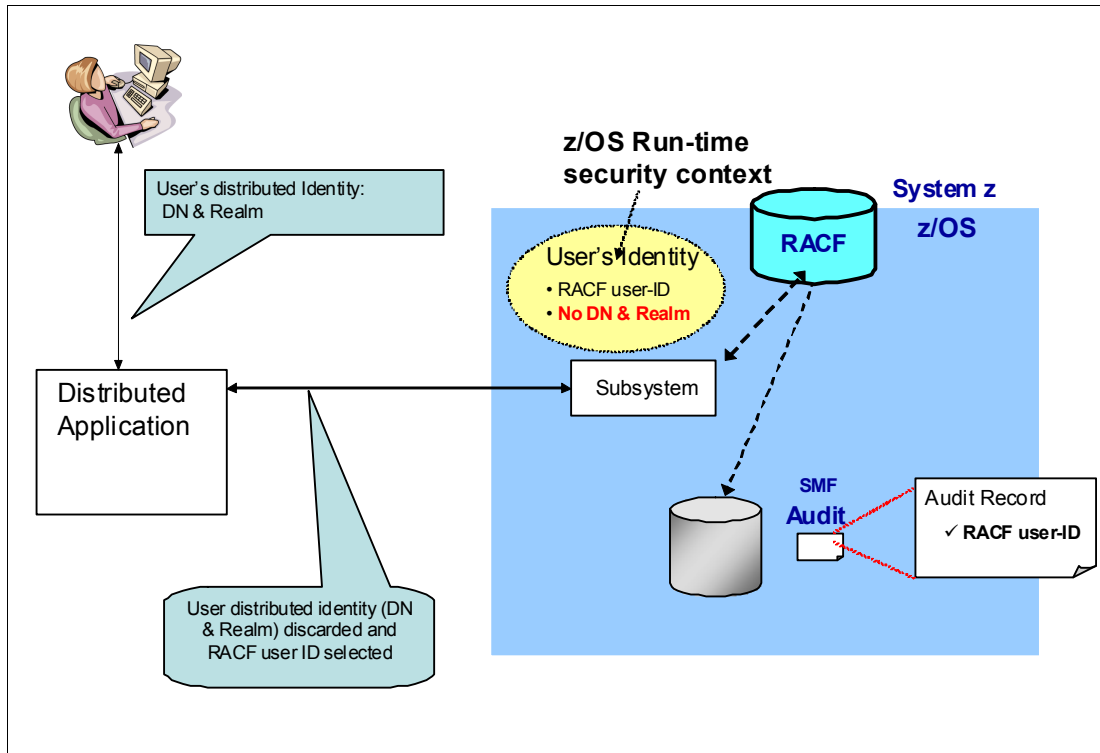


Figure 1-1 Without identity propagation

Identity propagation is the capability whereby a non z/OS identity, a *distributed identity*, is propagated into the z/OS environment and is then:

- ▶ Used to provide credentials for authorization by being mapped to an existing RACF user ID
- ▶ Available throughout the z/OS Sysplex for auditing and reporting

The distributed identity of the currently authenticated user is determined by the distributed application and passed along to the receiving subsystem on z/OS. The subsystem then uses a new form of RACROUTE VERIFY or RACF Callable Services that passes the distributed identity to RACF. A search of mappings is done and a runtime security context is built for the mapped RACF user ID, or the logon is failed if no mapping is found. The distributed identity is tracked within RACF, so now the user identity is not just a RACF user ID but also includes their distributed identity. Both the z/OS identity and the distributed identity will now be included in any subsequent audit records written by RACF. Figure 1-2 shows a conceptual overview of identity propagation.

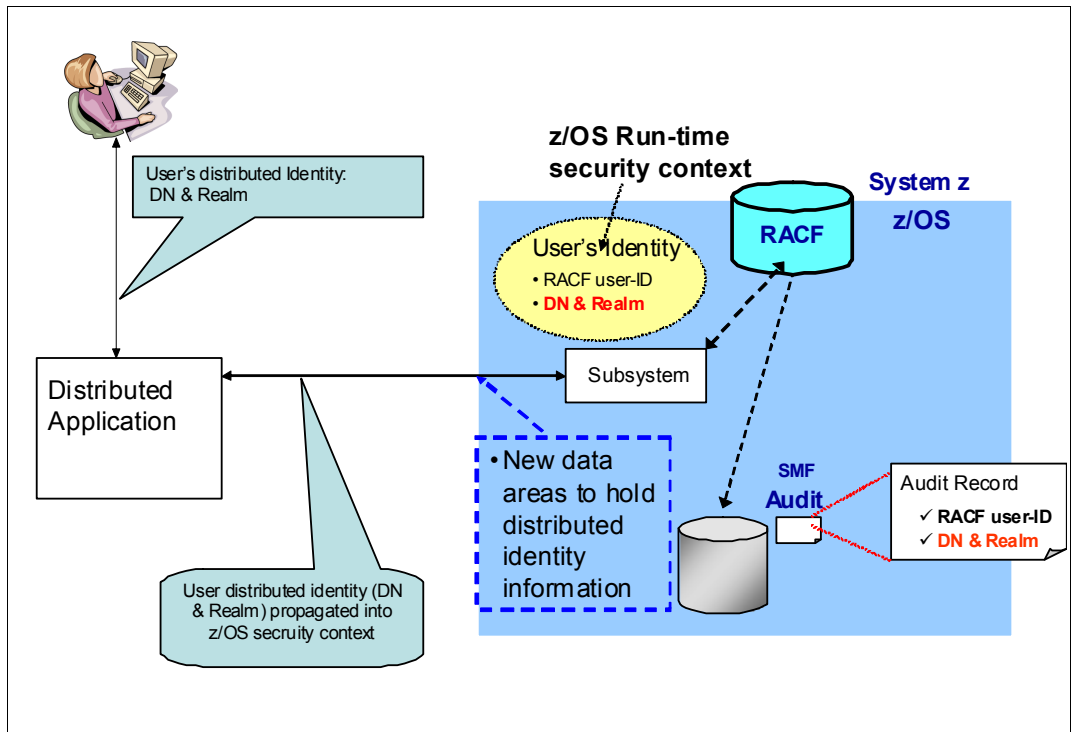


Figure 1-2 Identity propagation - Conceptual overview

Note that for the purposes of our discussion we consider the relevant parts of a distributed identity to consist of two components:

- ▶ Distinguished name (DN)
- ▶ Realm, or security database, known in RACF as *registry*

See 2.1.1, “Authentication, distinguished name, and registry” on page 9, for more information about distinguished name and registry.

1.2 Why identity propagation is important

Currently, in many z/OS environments, there are distributed applications that assert their identity in the form of a shared or generic RACF user ID that does not represent the true end user running the application. Consequently, neither the z/OS security server nor the z/OS audit trail can distinguish between the many users of that application, since we have lost accountability. However, now with identity propagation, the audit trail is complete and reflects both the RACF user ID and the distributed identity.

1.2.1 Complete end-to-end audit trail

It is not uncommon to observe distributed systems whose z/OS workload is run under a generic or shared user ID representing the application. Individual accountability is lost in this situation and we do not have full *end-to-end* security access controls, and the z/OS audit trail does not reflect the true end user's identity from the distributed system.

With identity propagation, the SMF audit trail will now contain the distributed identity, composed of a distinguished name and registry identification, along with the standard RACF user ID, providing enhanced accountability/auditing.

This distributed identity has been propagated from the distributed environment into the z/OS subsystem and passed to RACF when initial user identification is required (RACROUTE TYPE=VERIFY) in that subsystem. Instead of using an authentication mechanism to validate the assertion of a provided user ID, the distributed identity is *mapped* to a RACF user ID based on filters established by the security administrator. Subsequent resource authorization checks are performed in the usual manner using this user ID, and if auditing is indicated, the SMF record will contain both RACF user ID and distributed identity.

This allows for full end-to-end accountability because the audit trail now also contains the distributed identity, even if the RACF user ID is shared in some fashion.

See Chapter 6, "Using SMF audit information to report on z/OS Identity Propagation" on page 55, for details of SMF-based reporting.

1.2.2 Management of z/OS identity

A key element of identity propagation is the mapping process that determines what RACF identity should be used for a given distributed identity. This mapping process eliminates the need to authenticate an asserted user ID. With knowledge of the end user's true distributed identity, the security administrator can determine the suitable RACF user ID to be used. This moves the responsibility and control of user ID assertion from the application administrators to the z/OS security administrators.

RACF provides a new command, RACMAP, that is used to manage the user ID filters that map a distributed identity to a RACF user ID. There are no changes necessary to management of resource access controls. They continue to be managed using user ID or group permissions to resource profiles.

See Chapter 4, "RACMAP function" on page 39, for details on RACMAP and Chapter 5, "Filter management" on page 49, for details on management of the mapping filters.

The mappings in RACF are keyed from two elements of the distributed identity:

- ▶ The distinguished name (DN)
- ▶ The authentication registry or realm

The distinguished name is further broken down into multiple relative distinguished names (RDNs) that become less specific as you traverse the DN from left to right. The mapping process attempts a match for the full DN, and if not found it then iteratively strips off the leftmost RDN@ and searches for a match on the remaining portion of the DN. If all RDNs have been exhausted before a match is found, then a logon failure occurs.

Both the DN and registry can be specified with an asterisk (*), enabling a security administrator to provide various combinations of mappings.

The security administrator could construct various mappings to take advantage of the iterative mapping process of the RDNs as listed in Table 1-1.

Table 1-1 Simple example of possible mapping scheme

USERDIDFILTER	RACF user ID	Description
uid=simon,ou=swg,o=ibm	ITSO0K02	One-to-one mapping for a specific user
ou=swg,o=ibm	ITSO	Many-to-one mapping for other users in swg/ibm
**	NORACMAP	Default mapping for all others

Note: The default entry pointing to NORACMAP is not a requirement. See 4.9, “Default RACMAP filter protection” on page 46, for details about how to set up a default mapping. You only need to do this if you do not want a logon failure distributed identity not found if an unknown distributed Identity and Realm attempts signon.

RACF is not sensitive to the case of the RDN name, but it is sensitive to the case of the RDN value. UID= and uid= behave the same way. However, =simon and =SIMON behave differently.

1.2.3 Elimination of the perception of z/OS identity as a weak link

Currently, z/OS can sometimes be perceived as a weak link in a chain, due to the RACF restriction of 8 bytes maximum for the user ID size. With identity propagation, it no longer matters, because the distributed identity is also available along with the RACF user ID. Rather than increasing the size of the RACF identity, it can now be mapped with the distributed identity. RACF imposes a restriction on the size of the distinguished name of 246 bytes, and for the registry name, 255 bytes.

1.2.4 Making the distributed identity available to an application program

In the CICS environment, a CICS program can retrieve a distributed identity and registry by using the EXEC CICS INQUIRE ASSOCIATION command. The CICS application can then perform suitable processing of this identity for its own audit trail purposes or for applying business logic.

Note: These fields are in UTF-8 format and might need to be translated to EBCDIC depending on their intended usage.



RACF and z/OS Identity Propagation

This chapter describes how subsystems within z/OS, such as CICS and other exploiters, communicate to RACF to exploit identity propagation. This chapter discusses the callable services and data structures provided by RACF to support the propagation of a distributed identity. It also describes the information flow and how this identity is treated.

In this chapter reference is made to the enhancements supplied by APARs OA34258 and OA34259.

2.1 Conceptual overview

Implementing z/OS Identity Propagation requires customizing the z/OS Security Server and the exploiting applications and requires planning to map the distributed user identities to RACF user IDs. A conceptual understanding of the technical elements involved in z/OS Identity Propagation will assist in this process. Figure 2-1 shows how a typical application exploits z/OS Identity Propagation.

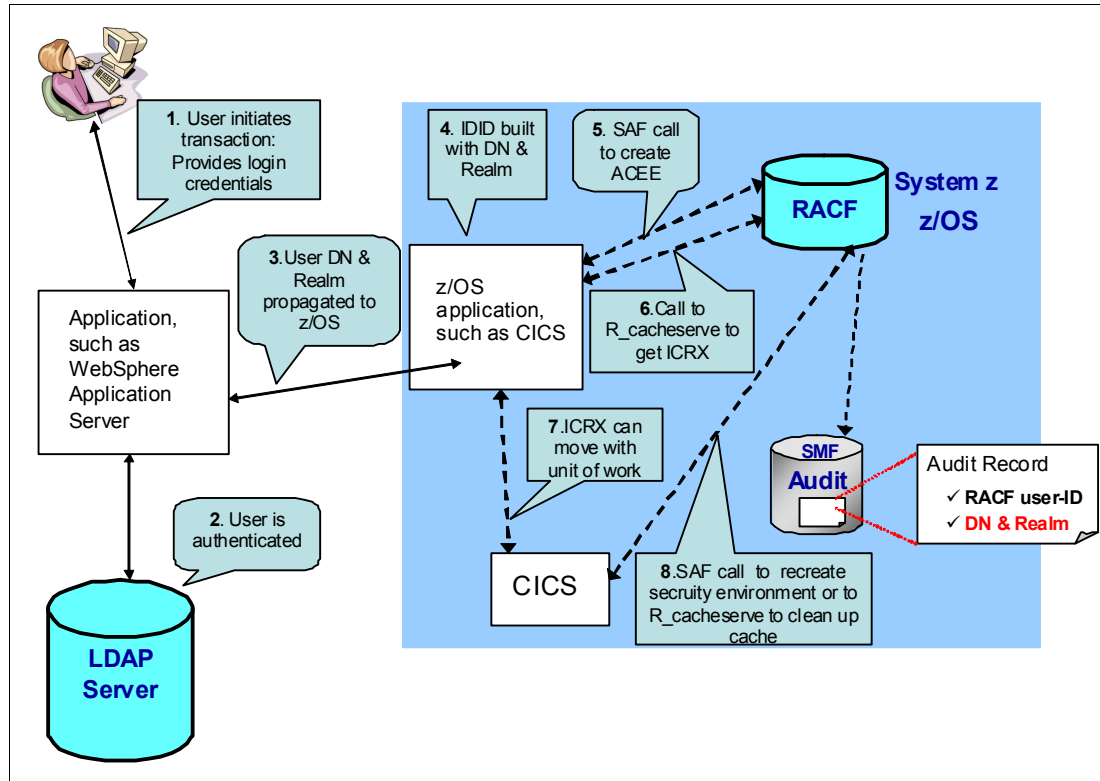


Figure 2-1 z/OS Identity Propagation steps

The steps shown in Figure 2-1 are summarized below, and further details are provided in the following sections:

1. A user initiates a transaction and provides login credentials to an application. The login information might be in the form of a DN, or the application might form the DN based on login credentials provided by the user.
2. The application authenticates the user using a security database. In this example it is an LDAP server configured for authentication,
3. Once authenticated, the application, if configured for identity propagation, either sends the required information (DN and registry) to the z/OS subsystem or constructs a Distributed Identity Data Structure (IDID), which is sent to the z/OS subsystem.
4. If the receiving subsystem on z/OS needs to construct a Distributed Identity Data Structure (IDID) using the DN and realm passed by the distributed application, it is done now.
5. The z/OS subsystem or application, in this example CICS, makes an API call to the security database (in this case RACF) to create the required z/OS runtime security environment, called an Accessor Environment Entry (ACEE). The IDID previously created is used in the call.

6. The application then makes a call to the R_cacheserv service, specifying the store function to get a Identity Context Reference Extended (ICRX). This data structure contains the Identity Context Reference (ICR) and the completed Identity Data Structure (IDID). The ICRX is also cached.
7. The ICRX obtained in step 6 can move with the unit of work through the system or sysplex. In this example it moves to another CICS region. This ICRX can be used to retrieve a record from the RACF local identity cache that contains information about the RACF user ID associated with the ICRX.
8. If the security environment has to be recreated, the application can make another API call using the ICRX obtained in step 6. If the security environment does not need to be recreated, the application can clean up the cache using the R_cacheserv service.

2.1.1 Authentication, distinguished name, and registry

Steps 1 and 2 of Figure 2-1 on page 8 illustrate a user initiating a transaction, of which part will execute on z/OS. In this example the user provides login information and is authenticated by the distributed application. This information (the DN of the user and the registry used for authentication (also know as realm or domain)) will be propagated to z/OS. The following sections provide additional details about the concept of DN and registry.

Distinguished name

The concept of a DN is fully documented in the X.500 series of standards published by the ITU Telecommunication Standardization Sector (ITU-T). In the context of our discussion, we simplify the concept to what is relevant for z/OS Identity Propagation.

A DN can be considered a series of delimited attribute type and attribute value pairs that attempts to uniquely identify an entity (such as a user of an application). Common attribute types are common name (CN), organizational unit (OU), organization (O), and country name (C). Example 2-1 is an example of a DN.

Example 2-1 A distinguished name

CN=Martina,C=Germany,OU=SWG,O=IBM

Each pair of attribute type and attribute value is considered a relative distinguished name (RDN). The distinguished name in Example 2-1 is composed of four RDNs, as listed in Table 2-1.

Table 2-1 RDNs

RDN attribute type name	RDN attribute value
CN	Martina
C	Germany
OU	SWG
O	IBM

The order of attribute types can be organized to reflect an organizational hierarchy, which results in a tree-like structure. For example, Figure 2-2 shows a simple organizational structure using RDNs.

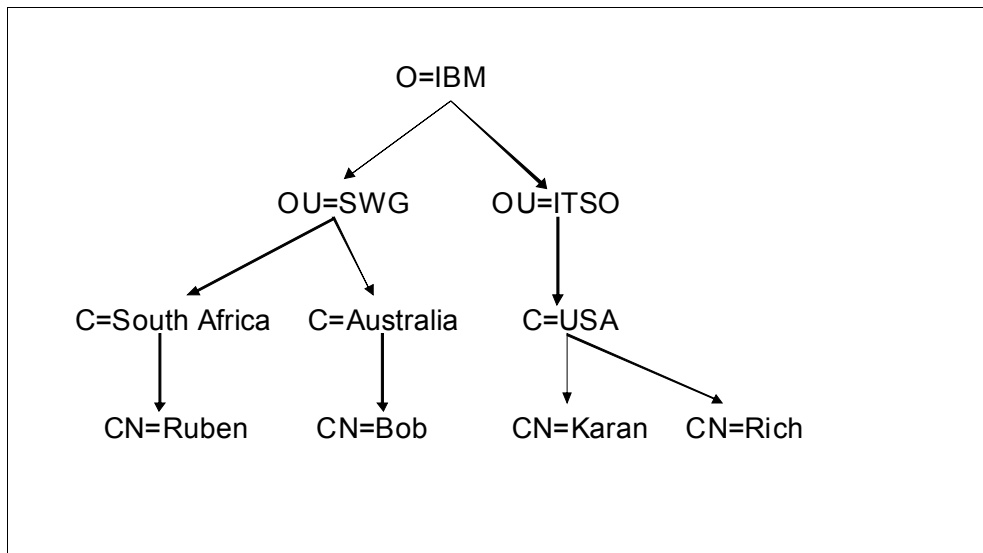


Figure 2-2 hierarchal structure

From Figure 2-2 the following four distinguished names are generated:

CN=Ruben,C=South Africa,OU=SWG,O=IBM

CN=Bob,C=Australia,OU=SWG,O=IBM

CN=Karan,C=USA,OU=ITSO,O=IBM

CN=Rich,C=USA,OU=ITSO,O=IBM

Registry or realm

In RACF, the database used for authentication of the distributed user is known as the registry (although in the distributed world it is commonly referred to as *realm* or *domain*). This value is the second piece of information required for RACF to map the distributed identity to a z/OS RACF user ID. The value of the registry is typically the security database used to authenticate the user, such as a Lightweight Directory Access Protocol (LDAP) server configured for authentication. This data is supplied by the application subsequent to authentication and provided to the component (which could be the application itself) that is creating the data structures required for the z/OS resource managers to use for identity propagation.

For example, in Figure 2-3, a user requests a transaction through WebSphere application server. The transaction will ultimately execute in a CICS region. In Figure 2-3 the user's login credentials are authenticated by the WebSphere Application Server via an LDAP database. Because this transaction has been configured in WebSphere Application Server for identity propagation, the user's distinguished name and the registry (in this case the LDAP server used for authentication) are passed on to z/OS.

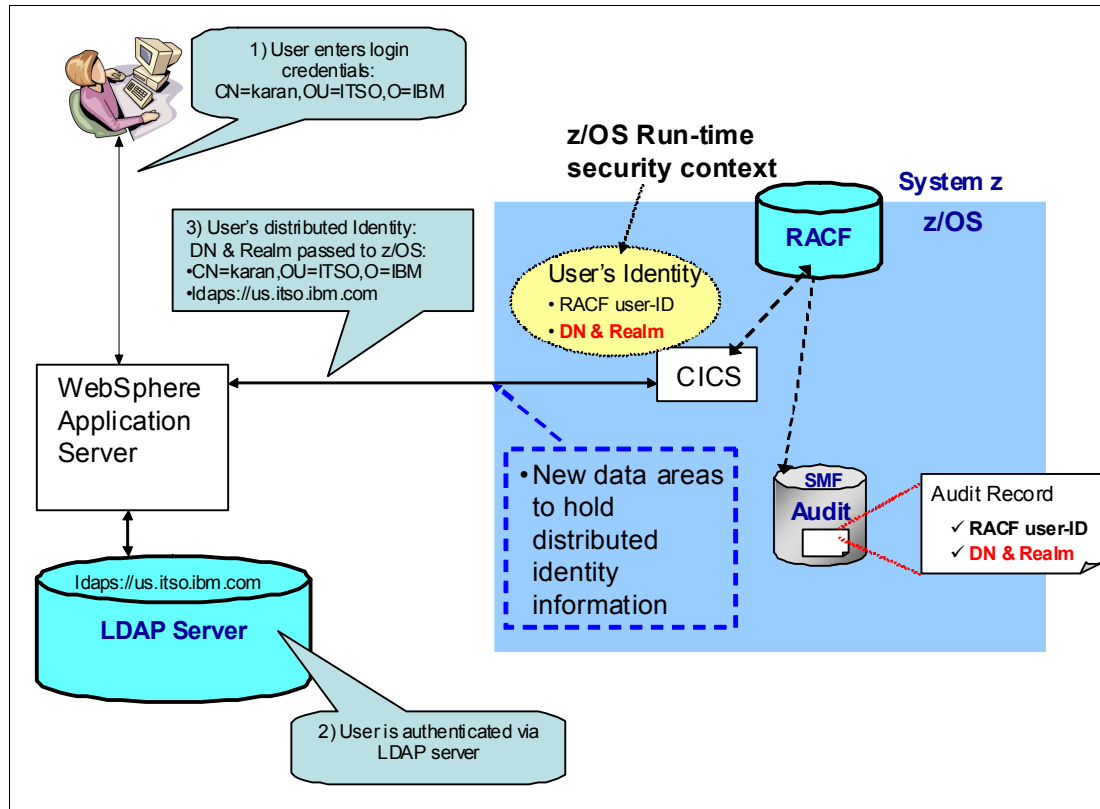


Figure 2-3 Registry

2.1.2 Distributed Identity Data Structure (IDID)

After a user has been authenticated, the distributed application passes the DN and realm to z/OS when configured for z/OS Identity Propagation. Depending on the software and hardware components involved in the particular identity propagation topology, the DN and realm are sent to a z/OS application, which builds a IDID, or the distributed application builds the IDID and sends the IDID to the z/OS application. This is illustrated as steps 3 and 4 in Figure 2-1 on page 8.

The IDID data structure contains the information needed by the z/OS Security Server to create an Accessor Environment Element (ACEE) containing the appropriate z/OS Identity Propagation references. This ACEE can be considered the z/OS runtime security context for a user ID. The IDID contains several sections, but from the caller's view the most important section is the z/OS section. In its structure it has two important fields:

- ▶ Distributed client end user's identity
- ▶ Registry's name

In addition, there is a section referred to as section 2 in the IDID data structure. That is reserved for the external security Manager (in our discussion, this is RACF). That section is

completed by RACF and present when a completed ICRX is delivered to the requesting application through a call to the R_cacheserv service.

2.1.3 SAF interfaces

After an IDID data structure has been created, it can be used in an API call to the security database to create a z/OS runtime security context. In step 5 of Figure 2-1 on page 8, the z/OS application has received or created a IDID data structure and now can issue a call to create the z/OS runtime security context.

Our application will use a SAF callable service (RACROUTE REQUEST=VERIFY or an initACEE) to create the runtime security context called the Accessor Environment Element (ACEE).

Variation can exist on the call depending on what is known at the time of the call:

- ▶ If the user ID is not to be determined by RACF, the calling application will supply to either RACROUTE REQUEST=VERIFY or the initACEE with the following input parameters: user ID and IDID parameters. If successful, an ACEE is supplied.
- ▶ If the user ID is to be determined by RACF, the calls using RACROUTE and initACEE are different. For the RACROUTE REQUEST=VERIFY, the application must construct an Extended Identity Context Reference (ICRX) data structure with an IDID holding the DN. In the case of initACEE, the application supplies the IDID data area as a parameter.

We now have an ACEE.

Note: For RACF to determine the RACF user ID to be associated with a distributed user ID, the IDIDMAP class must be active and RACLISTed. In addition, mapping filters must be defined using the RACMAP command. See Chapter 4, “RACMAP function” on page 39, for details on RACMAP and filters.

Extended Identity Context Reference (ICRX)

As pointed out above, depending on the call used, an application might have to create an ICRX data structure when requesting that an ACEE be built.

When specifying an ICRX as a parameter for the RACROUTE REQUEST=VERIFY call to request creation of an ACEE, the actions that the VERIFY logic will take depend on what information is present in the ICRX. Specifically, if an Identity Context Reference (ICR) section exists, then that ICR will be used during processing. This is usually the case when the security environment has to be recreated (an ACEE has been previously constructed and an ICRX generated).

Otherwise, if no ICR exists, which we assume is the case in our example for step 5 in Figure 2-1 on page 8, the following action is taken: If the ICRX does not contain an ICR, and the IDIDMAP class is active and RACLISTed. RACROUTE REQUEST=VERIFY processing attempts to map to a RACF user ID using the distributed identity information in the IDID and mapping filters previously defined using the RACMAP command. If the information in the IDID does not map to a RACF user ID, the RACROUTE REQUEST=VERIFY fails and returns a user not defined message.

If an ACEE is successfully created, the ACEE points to a copy of the IDID information from the ICRX, and it is used in auditing.

2.1.4 R_cacheserv and caching

After an ACEE has been built, the application invokes the R_cacheserv service and requests a STORE function as an action:

- ▶ If the application is in supervisor state and supplied the ACEE as a parameter on the R_cacheserv call, then this ACEE will be stored in this local identity context cache.
- ▶ Otherwise, the ACEE pointed to by the Task Control Block (TCB) or the Address Space Extension Block (ASXB) is placed in the cache.

Apart from the caching, the R_cacheserv outputs a complete ICRX. A complete ICRX contains:

- ▶ Identity Context Reference (ICR): Information to locate an item in the local identity context cache
- ▶ IDID with its section 2 complete with RACF information

The ICRX can be used by the application to issue further R_cacheserv calls to retrieve or remove a record from the cache. The record stored in the cache contains pertinent information about the RACF user ID associated with the ICRX.

This ICRX can now move through the system or sysplex with this unit of work. With the new enhancements supplied by APARs OA34258 and OA34259, this callable service provides support for subsystem callers such as CICS to create reusable ICRX objects.

2.1.5 Recreating a security environment or cleaning up a cache

When an application needs to re-establish the security environment, it invokes the RACROUTE REQUEST=VERIFY and supplies the complete ICRX.

If the application wants to clean up the local identity context cache it invokes the R_cacheserv service requesting a removal (function code 7, option 3). The complete ICRX is an input to the R_cacheserv call.

2.2 Internal logic within SAF call

Let us go a little deeper into the SAF calls mentioned in 2.1.3, "SAF interfaces" on page 12.

After an application has an IDID data structure created it has two methods for requesting the creation of an ACEE:

- ▶ RACROUTE REQUEST=VERIFY macro
- ▶ initACEE service

Furthermore, the application then has an additional choice to make that will affect what information it needs to provide to either call:

- ▶ Do not have RACF derive the RACF user ID. Instead, provide the RACF user ID. In this case RACF will not search the IDIDMAP class to find a profile.
- ▶ Have RACF derive the RACF user ID. RACF will search the IDIDMAP class for a matching profile.

Table 2-2 provides a simplified combination of choices and required data.

Table 2-2 Application choices when requesting ACEE creation

Action	RACF to derive user ID	IDID supplied	ICRX supplied	RACF user ID provided
RACROUTE REQUEST=VERIFY	Yes	Yes	Yes	No
RACROUTE REQUEST=VERIFY	No	Yes	No	Yes
initACEE	Yes	Yes	No	No
initACEE	No	Yes	No	Yes

The R_cacheserv service can be used to rebuild, retrieve, or remove a record from the cache. The following sections provide more details on these calls.

InitACEE decision logic

If the function code parameter indicates that an ACEE is to be created, and no user ID parameter is specified, and the IDIDMAP class is active and RACLISTed, then information in the IDID is used to determine a RACF user ID.

If the IDIDMAP class is inactive or the information provided in the IDID does not map to a RACF user ID, the initACEE service fails.

If a RACF_userid parameter is specified on the initACEE call and the IDID_area parameter can supply the name of an IDID to be associated with the ACEE, then the IDIDMAP class is not referred to.

If the IDID_area parameter is specified, the distributed identity information in the IDID should have been previously authenticated. If an IDID is supplied and an ACEE is successfully created, the ACEE will point to a copy of the IDID, and it will subsequently be used in auditing.

If a ACEE is successfully created, then the application should call the R_cacheserv to cache the ACEE as an ENVR object in the local identity context cache.

RACROUTE REQUEST=VERIFY decision logic

Variation on the call to RACROUTE REQUEST=VERIFY exists depending on whether an ICR exists.

When the ICRX contains an identity context reference (ICR), on the RACROUTE REQUEST=VERIFY, ENVIR=CREATE request, VERIFY uses it to determine a RACF user ID.

First, RACF attempts to resolve the ICR from the local identity context cache using the R_cacheserv callable service. RACF continues according to one of the following cases:

- ▶ If the ICR is resolved.

VERIFY retrieves an ENVR object for the user from the local identity context cache. This is used to create an ACEE for the caller. The IDID within the ICRX is ignored and reverification of information is not performed.

Note: When creating an ACEE using an ENVR object, the ENVR object might already contain an IDID.

- ▶ The ICR is not resolved.

If section 2 of the IDID, which is reserved for exclusive use by the External Security Manager, specifies a specific user ID, then VERIFY processing continues with this user ID and other security-relevant information within section 2 of the IDID.

If section 2 of the IDID does not exist or does not specify a RACF user ID, RACROUTE REQUEST=VERIFY fails the request and returns 'user not defined'.

Note: R_cacheserv attempts to resolve the ICR using the local identity context cache and also other relevant identity context caches that it can reach through RACF sysplex communication. Only ICRs that are created by an R_cacheserv store function are supported. See *z/OS Security Server RACF System Programmer's Guide, SA22-7681*, for more information.

Table 2-3 describes the logic flow when no ICR exists.

Table 2-3 Logic flow when no ICR exists

When the ICRX does not contain an ICR	
If the IDIDMAP class is active and RACLISTed, RACROUTE REQUEST=VERIFY processing attempts to map to a RACF user ID using the distributed identity information in the IDID and mapping filters previously defined using the RACMAP command.	If a RACF user ID is not determined (that is, data in the IDID does not map to a RACF user ID), the RACROUTE REQUEST=VERIFY fails and returns a user not defined message.
	If a RACF user ID is determined, RACROUTE REQUEST=VERIFY processing continues with this user ID, and PASSCHK=NO is assumed. All other supplied parameters are used. If an ACEE is successfully created, the ACEE points to a copy of the IDID information from the ICRX, and it is used in auditing.

More information

For more information see the following resources:

- ▶ For information about the RACROUTE macro refer to *z/OS Security Server RACROUTE Macro Reference, SA22-7692*.
- ▶ For information about the initACEE and R_cacheserv callable services refer to *z/OS Security Server RACF callable Services, SA22-7691*.
- ▶ The ACEE, IDID, and ICRX data areas are covered in *z/OS Security Server RACF Data Areas, SA22-7680*.

2.3 Impact on distributed identity data cached in RACF by z/OS subsystems

The cache referred to in 2.1.4, “R_cacheserv and caching” on page 13, is the local identity context cache that is communicated with by the R_cacheserv callable service. When an application invokes R_cacheserv with a store request, a record is stored in this cache reflecting a current in-force ACEE. A completed ICRX is returned to the caller on a successful store. The importance for z/OS Identity Propagation is that the returned ICRX contains an ICR that can be used by RACF to reference the cached ACEE from any system within the RACF Sysplex communication group, so the cache has a sysplex-scope.

We need to consider how to handle the circumstance in which an administrative action occurs on RACF user profiles. This might imply an already cached ACEE in the form of an ENVR object for the affected user to be deemed obsolete because of changes to the user's permission to access protected resources.

How could this happen? The user RACF profile is changed by one of the following RACF commands:

- ▶ CONNECT
- ▶ REMOVE
- ▶ ALTUSER with the REVOKE parameter

RACF will make a request to the Event Notification Facility (ENF) to send a signal to all the subsystems listening to it that an event 71 has occurred. The ENF facility can be considered a broadcast mechanism that allows an authorized program to listen for the occurrence of a specific system event. ENF event code 71 indicates that a RACF command has affected a user's group connections, which might change the resource authorization of a user. The commands triggering this event are listed above. This notification is important for z/OS Identity Propagation because a change in a user's authorization will affect any relevant cached records (that is, these records must be flushed and the record rebuilt).

You must remember that not all RACF administrative actions will request an ENF notification (for example, a password change does not require a cached ENVR object to become obsolete). Figure 2-4 outlines the process flow from RACF to a subsystem when a change requires a ACEE replacement.

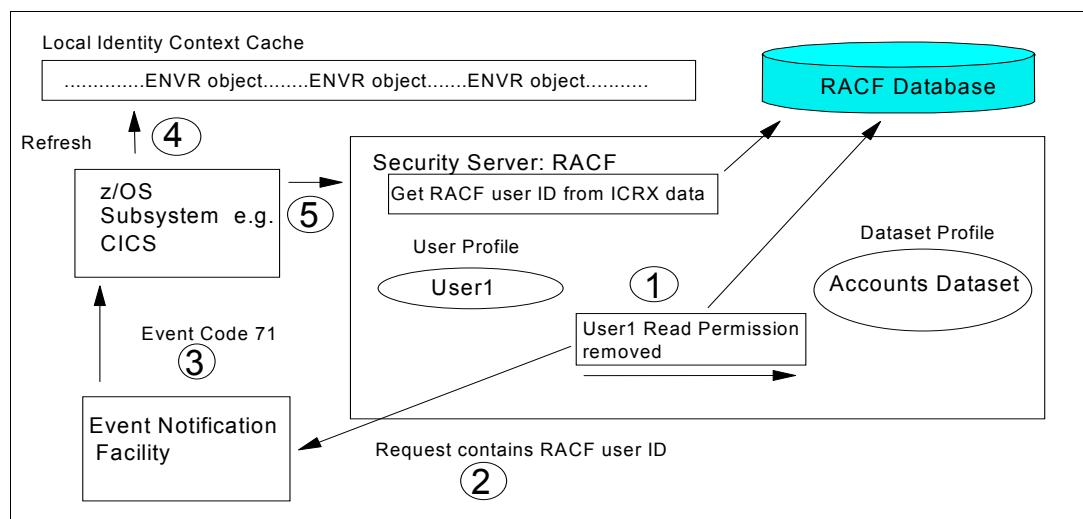


Figure 2-4 Process flow from RACF to a subsystem when a change requires a ACEE replacement

A change in the status of a RACF user ID is shown in position 1 in Figure 2-4 on page 16.

RACF issues an ENF 71 event code, and the subsystem such as CICS is notified immediately, overriding any setting that you specified in the USRDELAY system initialization parameter. This action is depicted at positions 2 and 3 in Figure 2-4 on page 16.

When CICS receives this signal from ENF it scans the local identity context cache for any ENVR objects. It checks for any RACF user IDs that have been impacted and deletes such ENVR objects from the cache. This action is depicted in position 4 in Figure 2-4 on page 16. It uses the R_cacheserv service to perform this action.

Position 5 covers the request to re-establish the application's security credentials by mapping the distributed identity to a RACF user ID creating an ACEE and then caching it.

Other systems can also listen for this signal and use it to clean up cached data. Table 2-4 sums up this description.

Table 2-4 Event notification signal 71 information

Description	Qualifier	Parameter list (passed to user exit)	Exit type	Cross-system capable
A RACF command has affected a user's group connections, which might affect his resource authorization. The user affected is in the parameter list in field IRR_ENF2USER.	The qualifier (QUAL) has the following format: BYTE1 X'80' CONNECT command X'40' REMOVE command X'20' ALTUSER REVOKE command BYTE2 - 4 Reserved	Mapped by IRRPENF2	Exit	Yes

A scan of the entire cache is needed in the worst possible case. However, this occurrence is relatively rare, and there should normally not be a significantly large number of entries in the cache as to cause a performance problem.

Scanning can be done asynchronously to normal transaction processing, so an impact on transaction processing performance is minimized.

Upon finding a matching ACEE cache entry, deletion of the entry occurs (or is otherwise marked as obsolete) and SAF-RACF R_cacheserv must be invoked, specifying a modified ICRX reflecting the ACEE entry, as shown in position 5 within Figure 2-4 on page 16.

The modified ICRX consists of IDID and RACF user ID information, both copied from the ACEE cache entry that is being deleted. This ICRX will contain no ICR data. SAF-RACF will use this ICRX to find any corresponding entries in its Sysplex global cache that are also obsolete and likewise delete them.

2.4 RACF database unload

For full technical details see *Security Server RACF Macros and Interfaces*, SA22-7682. The following discussion refers to new and changed records.

The User Associated Distributed Mappings Record (0209)

This defines the IDIDMAP class profile name associated with this user ID. This outlined in Table 2-5.

Table 2-5 Database Unload Record 0209

User Associated Distributed Mappings Record Defines the mappings record associated with this user ID				
Field name	Type	Position		Comments
USDMAP_RECORD_TYPE	Integer	1	4	Record type of the User Associated Distributed Mappings Record (0209)
USDMAP_NAME	Character	6	13	User ID as taken from the profile name
USDMAP_LABEL	Character	15	46	The label associated with this mapping
USDMAP_MAP_NAME *	Character	48	293	The name of the IDIDMAP profile associated with this user

* USDMAP_MAP_NAME is stored in the RACF database as UTF-8. The Database Unload Utility will translate to EBCDIC if possible. This is done to make these output records more readable. Should the translation not be possible, then it puts out hex values.

It should be noted that the same will apply to field GRBD_NAME in the General Resource Basic Data Record (0500) when the GRBD_CLASS_NAME is IDIDMAP.

The General Resource Distributed Identity Mapping Data Record (0509)

This output record defines the information used to create the mapping described by this IDIDMAP class profile and identifies the associated user ID (Table 2-6).

Table 2-6 Database Unload record 0509

General Resource Distributed Identity Mapping Data Record Defines the RACF user ID associated with an IDIDMAP profile				
Field name	Type	Position		Comments
GRDMAP_RECORD_TYPE	Integer	1	4	Record Type of the General Resource Distributed Identity Mapping Data Record (0509)
GRDMAP_NAME *	Character	6	251	General resource name as taken from the profile name
GRDMAP_CLASS_NAME	Character	253	260	Name of the class to which the general resource profile belongs
GRDMAP_LABEL	Character	262	293	The label associated with this mapping
GRDMAP_USER	Character	295	302	The RACF user ID associated with this mapping
GRDMAP_DIDREG *	Character	304	558	The registry name value associated with this mapping

* GRDMAP_NAME and GRDMAP_DIDREG are stored in the RACF database as UTF-8. The Database Unload Utility will translate them to EBCDIC if possible. This is done to make these output records more readable. Should the translation not be possible, it puts out hex values.

2.5 RACF messages

Message ICH408I has additional text available to it to describe an occurrence when a supplied distributed identity is not found (Figure 2-5).

```

ICH408I USER(STC      ) GROUP(TSO      ) NAME(STARTED TASK      )
DISTRIBUTED IDENTITY IS NOT DEFINED:
uid=martina,ou=swg,o=ibm wtsc58.itso.ibm.com:389
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.
  
```

Figure 2-5 z/OS Console showing ICH408I message

We must also consider the situation in which a RACF user is to be deleted. The user cannot be delete if there are any distributed identity mapping profiles still in existence for this particular user.

A new DELUSER message is issued if a related IDIDMAP profile is found when deleting a use (Figure 2-6).

```

ICH04018I ISTOUSER cannot be deleted. Distributed identity mapping profiles are
associated with this user.
  
```

Figure 2-6 Message showing a failure reason for a delete on a RACF user

2.6 RACF templates

The RACF database contains records, and the format of those records is under the control of database templates. These templates map how profiles are written to the RACF database.

To refer to documentation of these templates look in *z/OS Security Server RACF Macros and Interfaces, SA22-7682*, and *z/OS Security Server RACROUTE Macro Reference, SA22-7692*.

Accordingly, support exists for identity propagation in the user template and the general template. The information in the templates provides details about how IDIDMAP profiles are mapped to RACF user IDs.

The template additions shown in Table 2-7 show fields to assist with identity propagation in the user template. The additions occur in the base segment of this template.

Table 2-7 User template additions for identity propagation in base segment

Field name	Field ID	Flag 1	Flag 2	Field length (decimal)	Default value	Type	Field description
....
DMAPCT	095	10	00	00000004	00		Number of IDIDMAP mapping profiles that specify this user ID.
DMAPLABL	096	80	00	00000000	00		Label associated with this mapping.

Field name	Field ID	Flag 1	Flag 2	Field length (decimal)	Default value	Type	Field description
DMAPNAME	097	80	00	00000000	00		Name of mapping profile. The names correspond to profiles in the IDIDMAP class.

The template additions shown in Table 2-8 show fields to assist with identity propagation in the general template. The additions occur in the base segment of this template.

Table 2-8 General template additions for identity propagation in base segment

Field name	Field ID	Flag 1	Flag 2	Field length (decimal)	Default value	Type	Field description
...
DIDCT	075	10	00	00000004	00		Number of names that correspond to this IDID profile
DIDLABL	076	80	00	00000000	00		Label associated with this IDIDMAP class profile mapping (matches DMAPLABL for user named by DIDUSER)
DIDUSER	077	80	00	00000008	00		User ID
DIDRNAME	078	80	00	00000000	00		Registry name (max of 255)

These template tables give us a good view of how they interact:

- ▶ We see that we keep a count of the number of mapped profiles for that particular user ID.
- ▶ A count of the number of names in use, so that we cannot delete an IDIDMAP profile if we have a non-zero count.
- ▶ The association between an IDIDMAP profile and a RACF user ID is made through a label, the DIDLABL for the IDIDMAP profile and DMAPLABL for the user ID.

2.7 RACF remove ID utility

This utility is designed to remove from the RACF database user IDs and group IDs that are deemed no longer required. How does this impact identity propagation? As we have seen, we cannot delete a user ID if there exists one or more IDIDMAP mapping profiles for that user ID.

After unloading the RACF database with the IRRDBU00 utility program, we can run the IRRRID00 (remove ID utility) program against this unloaded database. That produces an output file containing RACF commands to delete IDs. In our case we should expect RACMAP DELMAP commands as well.

The JCL in Figure 2-7 was run to produce commands that would remove IDs relating to the list of RACF user IDs in the SYSIN file.

```
//REMOVE EXEC PGM=IRRRID00,REGION=OM
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SORTOUT DD UNIT=SYSALLDA,SPACE=(CYL,(15,0))
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(15,0))
//INDD DD DISP=SHR,DSN=CKRU.RACF.DBUNLOAD
//OUTDD DD DSN=ITSO.RACF.REMOVE.CLIST,
// SPACE=(CYL,(30,5),RLSE),UNIT=SYSALLDA,
// DCB=(LRECL=259,RECFM=VB,BLKSIZE=0),
// DISP=(NEW,CATLG,KEEP)
//SYSIN DD *
SWGAU
SWGDE
/*
```

Figure 2-7 Sample JCL to produce a list of RACF commands to remove IDs

If we examine the ITSO.RACF.REMOVE.CLIST dataset we see what this utility produces when there are associated mapping profiles for the selected users (Figure 2-8).

```
/******
/* The following commands delete profiles. You must review */
/* these commands, editing them if necessary, and then remove */
/* the EXIT statement to allow the execution of the commands. */
/******
EXIT
RACMAP ID(SWGAU) DELMAP(LABEL('LABEL00000001'))
RACMAP ID(SWGDE) DELMAP(LABEL('LABEL00000002'))
DELUSER SWGAU
DELUSER SWGDE
/******
/* IRRRID00 has successfully completed */
/******
```

Figure 2-8 Output from IRRRID00 showing RACF commands to delete IDs

2.8 Supplied samples

We have a number of supplied samples in SYS1.SAMPLIB with z/OS that have changes to meet the requirements of identity propagation, which we discuss in this section.

2.8.1 IKJTSO00 in RACPARM

If you invoke the RACF command RACMAP from with ISPF or TSO, then this command should exist in the TSO/E APF-AUTHORIZED COMMAND AND PROGRAM TABLES. A

sample is shown in the RACPARM member in the SYS1.SAMPLIB. Refer to the label IKJTSO00 in this member for an example of the RACMAP definition (Figure 2-9).

AUTHCMD NAMES(/* AUTHORIZED COMMANDS */	+
. . .			
LU	LISTUSER	/* RACF 5740-XXH */	+
RACDCERT		/* RACF 5645-001 */	+
RACMAP		/* RACF 5740-XXH */	+
RACPRIV		/* RACF 5740-XXH */	+
RALT	RALTER	/* RACF 5740-XXH */	+
	RACLINK	/* RACF 5695-039 */	+
RDEF	RDEFINE	/* RACF 5740-XXH */	+
RDEL	RDELETE	/* RACF 5740-XXH */	+

Figure 2-9 A portion of the AUTHCMD entries from within RACPARM member in SYS1.SAMPLIB

2.8.2 Sample DB2 members

RACF supplies two sample members to take unloaded SMF data and then load into DB2 tables.

First, we have member IRRADUTB in SYS1.SAMPLIB, which defines the table space and tables used to contain the output of the RACF SMF data unload utility (IRRADU00). This is in the form of sample data definition language (DDL) statements that can be used to create the tablespace IRRADU00 in the database RACFDB2. For identity propagation a new table is created and columns defined for distributed identity data:

- ▶ Add (CREATE) table USER01.RACMAP.
- ▶ Define columns for UTF-8 and EBCDIC versions of IDID data.

In the example in Figure 2-10 we only show the DDL statements for the USER01.RACMAP table. Similar DDL statements exist for other tables in this sample member to hold identity data.

```

-----
--          RACMAP          --
-----
CREATE TABLE USER01.RACMAP(
    RACM_EVENT_TYPE          CHAR(8),
    ...
    RACM_IDID_USER_UTF8      VARCHAR(246) FOR BIT DATA,
    RACM_IDID_USER_EBCDIC    VARCHAR(738),
    RACM_IDID_REG_UTF8       VARCHAR(255) FOR BIT DATA,
    RACM_IDID_REG_EBCDIC     VARCHAR(765)
) IN RACFDB2.IRRADU00;

```

Figure 2-10 Portion of DDL statements to define a table and columns relating to a distributed identity

Member IRRADULD within SYS1.SAMPLIB provides statements to load into DB2 all the record types produced by the SMF unload utility. These are sample DB2 load utility statements that can be used to load the output of IRRADU00 into these tables.

The member can be edited to only load the required record types. These load statements now cater for the distributed identity and also load it in both UTF-8 and EBCDIC format. This is demonstrated in the Figure 2-11, as we show portions of the load statements for JOBINIT and ACCESS SMF records.

```

LOAD DATA
  INDDN IRRADU00
  RESUME YES
  LOG NO
  INTO TABLE USER01.JOBINIT
  WHEN(1:8)='JOBINIT' (
    INIT_EVENT_TYPE      POSITION(1:8)          CHAR(8),
  ...
    INIT_IDID_USER_UTF8  POSITION(2532:2777)    CHAR(246),
    INIT_IDID_USER_EBCDIC POSITION(2779:3516)    CHAR(738),
    INIT_IDID_REG_UTF8   POSITION(3518:3772)    CHAR(255),
    INIT_IDID_REG_EBCDIC POSITION(3774:4538)    CHAR(765)
  )

  INTO TABLE USER01.ACCESS
  WHEN(1:8)='ACCESS' (
    ACC_EVENT_TYPE      POSITION(1:8)          CHAR(8),
  ...
    ACC_IDID_USER_UTF8  POSITION(3346:3591)    CHAR(246),
    ACC_IDID_USER_EBCDIC POSITION(3593:4330)    CHAR(738),
    ACC_IDID_REG_UTF8   POSITION(4332:4586)    CHAR(255),
    ACC_IDID_REG_EBCDIC POSITION(4588:5352)    CHAR(765)
  )

```

Figure 2-11 Selected load statements relating to identity propagation in IRRADULD

2.8.3 XML schema document

Member IRRSCHEM is provided in SYS1.SAMPLIB. It defines the XML grammar used to validate the XML instance documents produced by the RACF SMF data unload utility (IRRADU00).

It is an XML schema document that defines the XML tag language, as highlighted in partial form in Figure 2-12.

```
<!-- ----->
<!-- Beginning of the security event log element definitions -->
<!-- ----->
  <xs:element name="securityEventLog">
    <xs:complexType>
      <xs:sequence>
        ...
          <xs:element name="ididReg" type="t_string1_1021"/>
          <xs:element name="ididUser" type="t_string1_985" />
        ...
      </xs:sequence>
    </xs:complexType>
  </xs:element> <!-- securityEventLog element -->
```

Figure 2-12 Portion of the SML schema document



z/OS Identity Propagation exploiters

This chapter describes the subsystems on z/OS Version 1 Release 11 exploiting identity propagation. This chapter discusses how to enable identity propagation on each of these subsystems, including:

- ▶ CICS and CICS TG
- ▶ CICS Web services
- ▶ DB2 10 for z/OS

3.1 CICS Transaction Gateway

The CICS Transaction Gateway (CICS TG) is a set of client and server software components that allow a remote client application to invoke services in a CICS region. The client application can be either a Java application or a non-Java application using either C, C++, COBOL, or COM interfaces (depending on the platform used).

When a Java application is used, the application can be any type of client (such as a servlet or an enterprise bean). In the JEE environment, the application is typically a servlet or enterprise bean that is deployed into a JEE application server such as WebSphere Application Server.

The CICS TG provides a set of JCA resource adapters that provide a simple and robust way of connecting a JEE application to a CICS application. Figure 3-1 shows how the CICS External Call Interface (ECI) resource adapter enables access to a CICS business logic program.

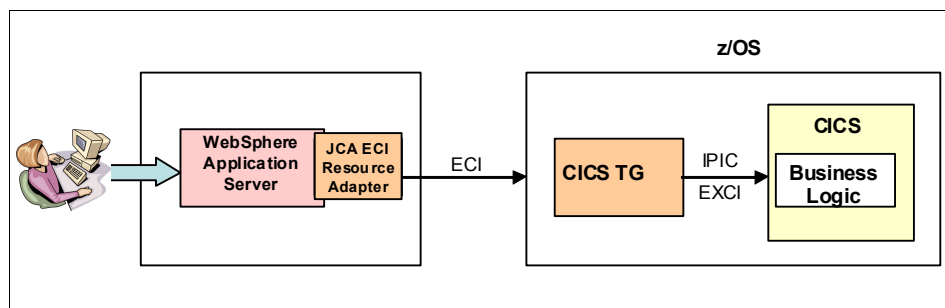


Figure 3-1 CICS Transaction Gateway

The CICS TG ECI classes are packaged with the ECI resource adapter and are used to pass the application request to the CICS TG. Rational® Application Developer can be used to create a Java bean to represent a COMMAREA formatted as COBOL types, with Java methods for getting and setting fields.

In Figure 3-1, the CICS TG daemon is deployed on z/OS. It can be configured to use either External CICS Interface (EXCI) or an IP interconnectivity (IPIC) to communicate with CICS.

3.1.1 JCA and security

The JCA defines a standard set of system-level contracts between a JEE application server and a resource adapter. These system-level contracts define the scope of the managed environment that the JEE application server provides for JCA components. One of the standard contracts is the security management contract that enables secure access to an EIS. Both container-managed sign-on (in which the JEE application server is responsible for flowing security context to the EIS) and component managed sign-on (in which the application is responsible for flowing security context to the EIS) are supported.

Container-managed security is suggested because it is good practice to separate the business logic of an application from qualities of service such as security.

3.1.2 CICS TG for z/OS security options

In Figure 3-1 on page 26, the Gateway daemon is the entry point to the System z platform in which the CICS system is running, so it is normal for the Gateway daemon to perform an authentication check for incoming ECI requests from clients.

The CICS TG for z/OS supports the following options for securing ECI requests:

- ▶ Basic authentication

The CICS TG can be configured to validate a user ID and password for each ECI request.

- ▶ Identity assertion

After the user has authenticated to WebSphere Application Server, the user's password is unlikely to be available to send to the Gateway daemon. In this case, the Gateway daemon and CICS can be configured to accept a user ID without a password.

This is a form of identity assertion, and therefore you should establish a trust relationship between the WebSphere application server, the Gateway daemon, and the CICS server.

- ▶ z/OS Identity Propagation

This is a unified security solution that provides additional accountability, which is achieved by passing a distributed identity to CICS instead of a user ID and password.

This is also a form of identity assertion, and therefore you should establish the required trust relationships (see "Trust" on page 27).

Note: z/OS Identity Propagation is supported with the CICS TG only when using ECI calls over IPIC connections to CICS.

- ▶ SSL/TLS

SSL/TLS can be used for confidentiality, data integrity, and optionally for X.509 certificate authentication.

Trust

For identity propagation with CICS TG, we suggest that the connection between the WebSphere application server and the Gateway daemon be configured as a trusted connection using one of the following methods:

- ▶ Using SSL client authentication to authenticate the application server to the Gateway daemon.
- ▶ Using a Virtual Private Network (VPN) or other network security configuration.
- ▶ Using a CICS TG security exit, which allows simple pre-configured rules to be set, ensuring that only specific application servers with a known key can connect.

A trust relationship should also be configured between the Gateway daemon and the CICS server. When an IPIC connection is used, trust can be established in one of the following ways:

- ▶ Using a pre-defined CICS IPCONN resource definition, which is a basic form of security that only allows a Gateway daemon configured with the correct name to connect to CICS.
- ▶ Using sysplex sockets, which can ensure that if non-SSL IPIC connections are used with CICS, then they must come from an IP stack on the same sysplex.

- ▶ Using NET ACCESS zones, which are RACF-based rules used by the IP stack that can prevent or allow preselected IP addresses from connecting to a given TCP/IP service in use by CICS.
- ▶ Using a firewall technology.

3.1.3 z/OS Identity Propagation support with CICS TG

Identity propagation provides a new security architecture for controlling identity assertion when connecting JEE applications to CICS. Identity propagation provides for a variety of identity assertion configurations when the WebSphere Application Server and CICS components are connected in a secure topology. Being in a secure topology stipulates that either the Gateway daemon and CICS region are located on the same sysplex or that they are connected by a client-authenticated SSL connection.

Figure 3-2 shows an overview of a CICS TG identity propagation scenario.

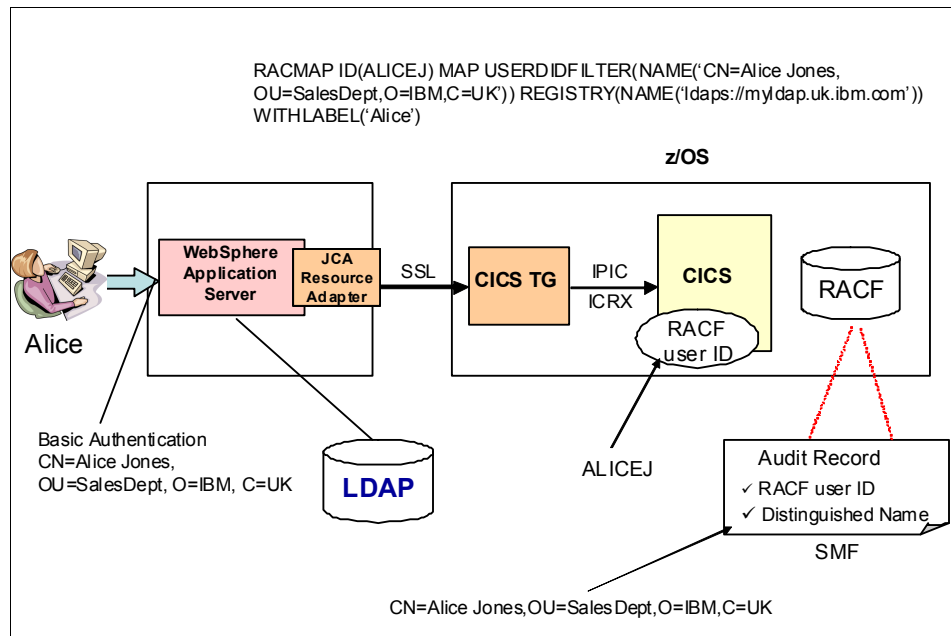


Figure 3-2 z/OS Identity Propagation with CICS TG

Figure 3-2 shows an employee (Alice) who logs on to the company’s WebSphere Application Server. After successful authentication, the WebSphere application makes a JCA call to CICS. The company has a requirement to authorize requests based on Alice’s RACF user ID.

The sequence of processing steps is:

1. Alice logs on to the WebSphere application server and authenticates with her distributed identity (CN=Alice Jones,OU=SalesDept,O=IBM,C=UK).
2. The JEE application is configured to use a CICS TG provided login module that attaches Alice’s distributed identity, in the form of DN, onto the outbound request to the CICS TG.
3. The SSL connection from the WebSphere Application Server to the Gateway daemon is client authenticated, thus establishing a trust between the two servers.
4. The Gateway daemon receives Alice’s DN and flows it as part of an Extended Identity Context Reference (ICRX) to CICS.

5. CICS receives the request from the Gateway daemon and uses the ICRX to identify the user.
6. CICS issues a RACROUTE REQUEST=VERIFY to map the ICRX into the RACF user ID ALICEJ.
7. The CICS task runs under the mapped RACF user ID (ALICEJ) but retains the association with the original distributed identity (CN=Alice Jones,OU=SalesDept,O=IBM,C=UK).

3.1.4 CICS resources used for configuring identity propagation with CICS TG

This section summarizes the CICS resources that a systems programmer uses to configure z/OS Identity Propagation with the CICS TG. The main CICS resource definitions to consider are:

► **TCPIPSERVICE**

A TCPIPSERVICE definition is required for IPIC connectivity from the Gateway daemon to CICS. It contains information about the port on which inbound requests are received, and whether any transport-based security mechanisms, for example, SSL/TLS, will be applied by CICS.

► **IPCONN**

An IPCONN resource defines the IPIC connection. To configure a CICS region to use a distributed identity flowed from the CICS TG requires that the USERAUTH attribute of the IPCONN resource definition specifies the value IDENTIFY.

For information about configuring identity propagation with the CICS TG, see Chapter 10, “Identity propagation using CICS Web services” on page 123.

3.2 CICS Web services

Application programs running in CICS TS V3 or later can participate in a heterogeneous web services environment as service requesters, service providers, or both, using either an HTTP transport or an MQ transport. Figure 3-3 shows a high-level view of the key components when CICS acts as a service provider.

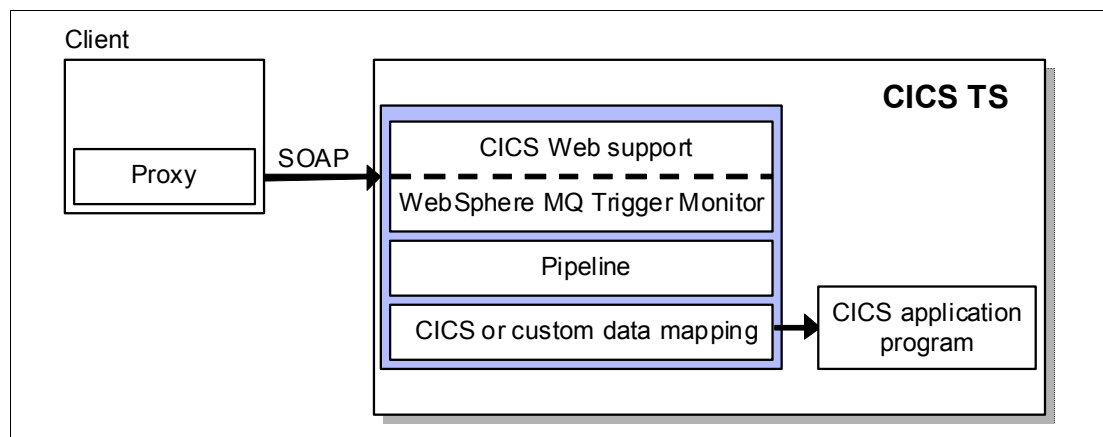


Figure 3-3 CICS Web services

A *message handler* in the pipeline (typically, a CICS-supplied SOAP message handler) removes the SOAP envelope from the inbound request and passes the SOAP body to the

data mapper function. The data mapper uses the CICS WEBSERVICE definition to locate the mapping information that it needs to map the inbound service request (XML) to a COMMAREA or a container. The output data from the CICS application program is converted into a SOAP message and sent back to the service requester.

The *data mapping* is generated either using the CICS-supplied utilities or Rational Application Developer for System z (RDz). CICS-supplied or custom-written message handlers are inserted into the pipeline in order to control processing of the request. For example, a security message handler can be used to change the security context of a request, that is, to change the RACF user ID under which the target request runs.

When CICS is a service requester, an application program sends a request, which is passed through a pipeline, to a target service provider. The response from the service provider is returned to the application program through the same pipeline.

3.2.1 CICS resources used for configuring web services

This section summarizes the CICS resources that a systems programmer uses to configure CICS Web services that are accessed over HTTP. For more detailed information about configuring CICS Web services, refer to the IBM Redbooks publication *Implementing CICS Web Services*, SG24-7206.

The main CICS resource definitions to consider when securing CICS Web services are:

► TCPIPSERVICE

A TCPIPSERVICE definition is required in a service provider that uses HTTP or HTTPS as transport. It contains information about the port on which inbound requests are received, and whether any transport-based security mechanisms will be applied by CICS. A TCPIPSERVICE definition is used to configure transport security.

► URIMAP

A URI mapping or URIMAP resource definition matches the URIs of web service requests. The URIMAP associates a URI for the request with a PIPELINE and WEBSERVICE resource that specifies the processing to be performed. You can use a URIMAP to specify:

- The name of the transaction that CICS uses for running the pipeline alias transaction (The default is CPIH.)
- The user ID under which the pipeline alias transaction runs

► PIPELINE

A PIPELINE resource definition provides information about the message handlers that act on a service request and on the response. The PIPELINE provides the name of a pipeline configuration file, and the pipeline configuration file specifies the list of message handlers.

To configure a pipeline to support SOAP message security, you can add the CICS-supplied security handler DFHWSSE1 to the pipeline configuration file. You can use the DFHWSSE1 message handler for a range of message security functions including validating XML signatures, XML encryption, and identity propagation.

► WEBSERVICE

A WEBSERVICE resource defines aspects of the runtime environment for a CICS application program deployed in a web services setting, where the mapping between application data structure and SOAP messages has been generated using the CICS Web services assistant.

Chapter 10, “Identity propagation using CICS Web services” on page 123, describes how we configured these CICS resources for our web services identity propagation scenario.

3.2.2 Securing CICS Web services

Different IT projects will have different security objectives. After the specific objectives are understood, you can consider two types of security mechanisms in a CICS Web services environment:

- ▶ Transport-based security

Transport-based security mechanisms such as SSL/TLS can be used to secure web services. SSL/TLS is a mature technology that has been optimized over a long period of time, and there are ways of optimizing performance such as persistent TCP/IP connections and the use of hardware acceleration for cryptographic processing.

You might choose to use transport-based security when there are no intermediaries, transport is only based on HTTP, or performance is your primary concern.

CICS transport security is configured by specifying security options in the TCPIP SERVICE resource definition.

- ▶ SOAP message security

SOAP message security using the WS-Security specifications provides a foundational set of SOAP message extensions for building secure web services by defining new elements to be used in the SOAP header. It specifies the use of security tokens, digital signatures, and XML encryption to protect and authenticate SOAP messages.

You might choose to use WS-Security (possibly in addition to transport-based security) when intermediaries are used. Security credentials that flow in the SOAP message can pass through any number of intermediaries. Furthermore, an intermediary might be able to provide an authentication service to CICS, such that the intermediary server authenticates the web service client and then flows an asserted identity to CICS.

CICS SOAP message security is configured by specifying the WS-Security security handler as part of the pipeline configuration file.

3.2.3 WebSphere DataPower

WebSphere DataPower® is a hardware solution that is well known for its security features and its high throughput in XML processing. It can be integrated in a DMZ and it can detect and reject XML attacks. The appliance can act as an XSL accelerator and transformation engine, be used as a Firewall and security device (authentication, authorization, encryption, and decryption messages), and also function as an Enterprise Service Bus (ESB). It can perform complex security checks without performance degradation.

WebSphere DataPower can be used in conjunction with CICS Web services to help secure the services and to offload expensive operations by processing the complex part of XML messages (such as an XML digital signature) at wirespeed (Figure 3-4).

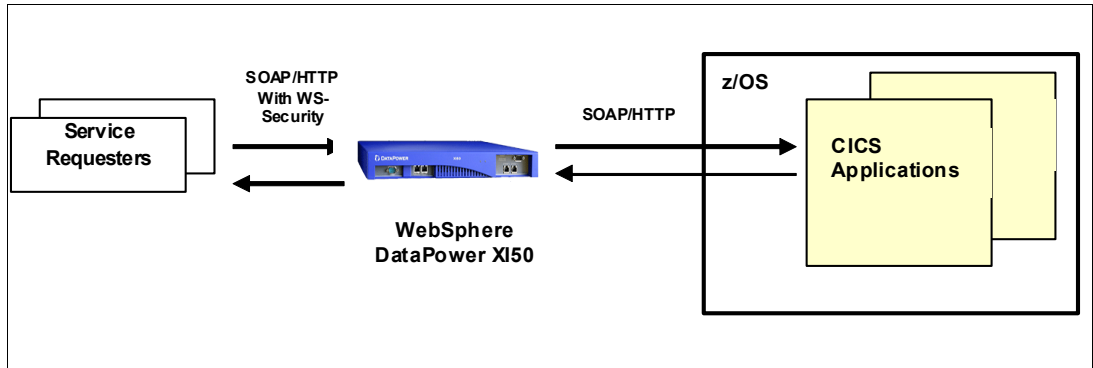


Figure 3-4 Using a WebSphere DataPower SOA Appliance with CICS Web services

3.2.4 Identity propagation with CICS Web services

WebSphere DataPower can be used to implement identity propagation with CICS Web services. Figure 3-5 shows how DataPower can be used to propagate a distributed identity to CICS so that it can be mapped to a RACF ID.

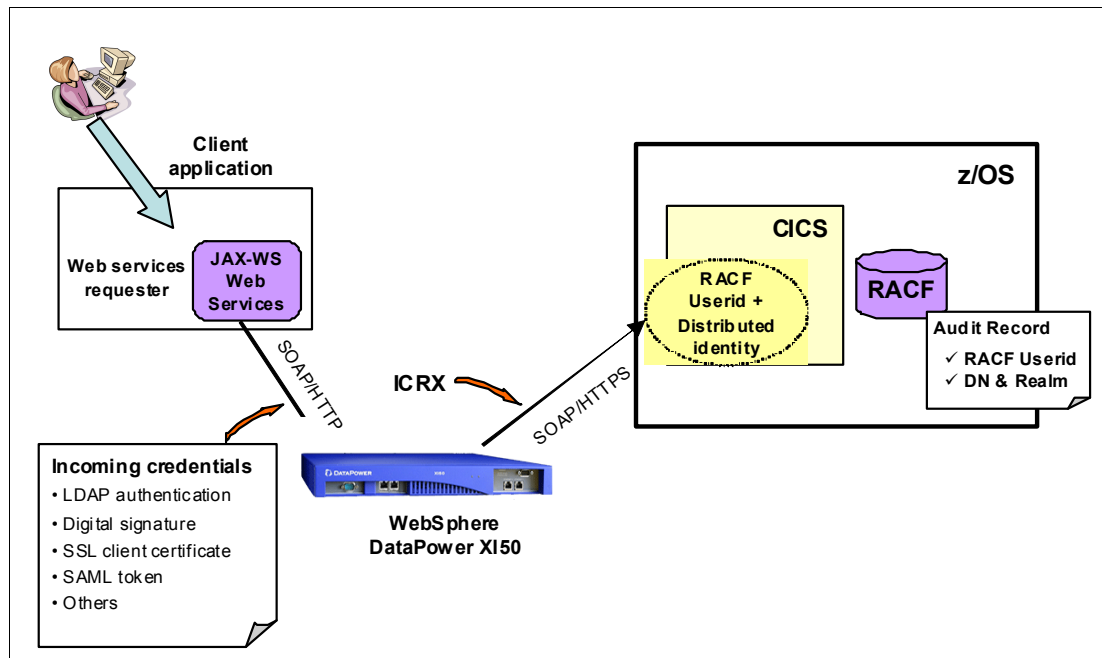


Figure 3-5 Identity propagation with CICS Web services

In Figure 3-5, the web service requester application connects to WebSphere DataPower using the SOAP protocol. WebSphere DataPower authenticates the credentials supplied by the distributed user and maps them to a z/OS ICRX identity token, which contains the distributed identity of the user.

WebSphere DataPower supports a wide range of authentication mechanisms, including:

- ▶ LDAP authentication
The requester is authenticated by an LDAP server.
- ▶ Digital signature
The requester is authenticated via the certificate passed as part of the <X509/> element of a digitally signed message.
- ▶ SSL client certificate
The requester is authenticated via its client SSL credentials.
- ▶ SAML token
The requester is authenticated by a SAML server or by a SAML assertion with a valid signature.

The SOAP message is forwarded to CICS over a trusted SSL connection with the ICRX identity token in a WS-Security header. Identity propagation is supported for CICS Web services when using WebSphere DataPower XI50 or XS40 and at a minimum level of firmware (Version 3.8). Any supported levels of WebSphere Application Server can be used as the Web services requester because the ICRX is created in DataPower based on the distributed identity provided by the application server.

Note: CICS Web services identity propagation is supported with CICS Transaction Server V4.1 with a set of enabling APARs:

- ▶ PK83741
- ▶ PK95579
- ▶ PM01622

APAR PK98426 is also required if you are using CICSplex® SM.

CICS receives the SOAP message from DataPower and passes the ICRX to RACF so that the client's identity can be mapped to a RACF user ID. The CICS task then runs under this RACF user ID, but retains the association with the original distributed identity. The advantage of this solution is that the original caller's identity is not lost. It is stored as an extension to the RACF identity.

Trust

For identity assertion with web services, the intermediary server should establish a *trust relationship* with the CICS region by authenticating itself and then by being recognized as a trusted partner of the CICS region. CICS supports two models for establishing this trust relationship:

- ▶ Trust token
The intermediary server sends a trust token to CICS.
- ▶ Blind trust
Trust is established at the transport level, for example, with SSL client authentication.

CICS Web services identity propagation support is normally used with the blind trust model, which has the advantage that the trust established between the intermediary server and CICS can be persistent. This can occur, for example, by using SSL persistent connections or a Virtual Private Network (VPN). It does not need to be re-established for each SOAP message. When using SSL client authentication to establish the trust relationship, the SSL certificate that WebSphere DataPower uses to identify itself can be associated with a RACF

user ID, and *surrogate user checking* can then be used to authorize this user ID to assert the RACF ID that is mapped from the distributed identity.

Note: A *surrogate* user is a RACF user ID that is authorized to act on behalf of another user (the original user).

For information about configuring identity propagation with CICS Web services and WebSphere DataPower, see Chapter 10, “Identity propagation using CICS Web services” on page 123.

3.3 DB2 10 for z/OS

This section describes DB2 for z/OS exploiting identity propagation and what is required to propagate the *distributed identity* through DB2. This section explains how distributed identities can be consistently mapped to a RACF user ID, which is then used for normal DB2 authentication.

From z/OS Security Server V1R11, the EIM function is fully integrated into the RACF database. As a result, you no longer have to provide the LDAP server and EIM domain controller infrastructure that is required for the DB2 EIM implementation. This function provides a more integrated approach to manage user IDs across the enterprise. DB2 10 for z/OS is exploiting identity propagation.

To exploit this feature, the application server must be connecting to DB2 10 for z/OS and z/OS Version 1.11 or later. Exploiters of identity propagation on DB2 for z/OS are remote systems, connecting using DRDA® and trusted connections within DB2. This chapter discusses how to enable identity propagation for DB2 for z/OS, including:

- ▶ DB2 10 for z/OS Trusted Context
- ▶ RACMAP
- ▶ SMF reporting
- ▶ Distinguished name not matching

A distributed identity filter is a mapping association between a RACF user ID and one or more distributed user identities. 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 on z/OS, use the RACF RACMAP command to associate (or map) a distributed user identity to a RACF user ID.

Trusted Context feature on DB2 10 for z/OS enables a trusted connection with a distributed system. After a trusted context is defined and initial trusted connection to the database server is made, the application server can use that database connection from a different user without a full reauthentication.

Figure 3-6 illustrates the entities involved with identity propagation.

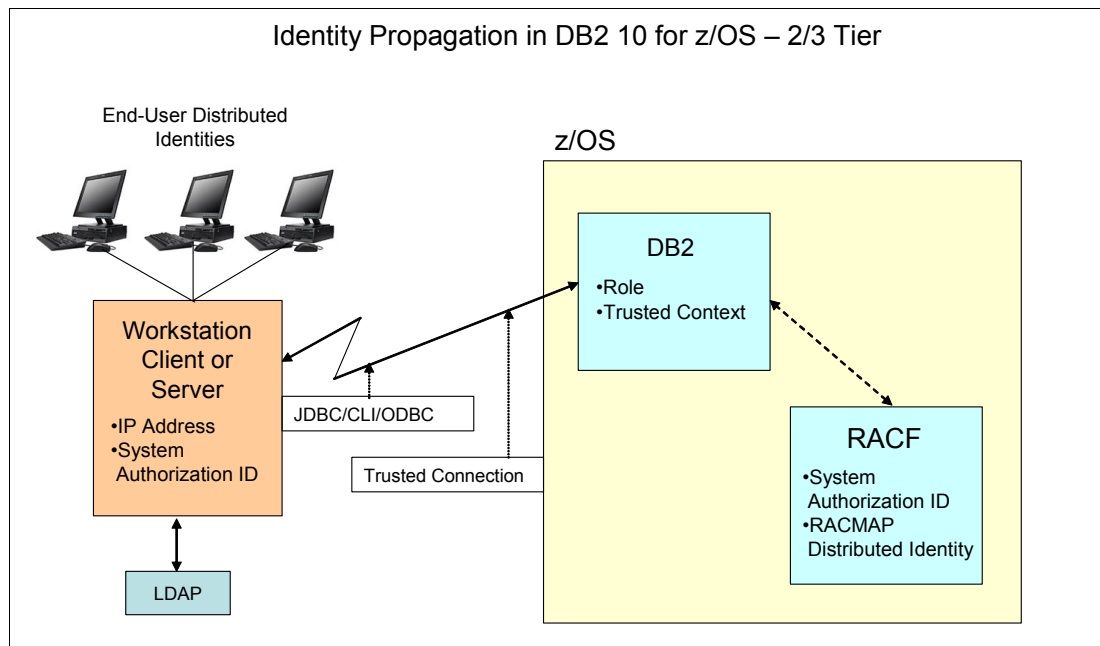


Figure 3-6 Different entities for DB2 10 for z/OS Identity Propagation

Figure 3-6 shows client workstations connecting to DB2 for z/OS using either JDBC or CLI/ODBC connections. This is using a trusted connection based on the trusted context created in DB2. Within RACF, mapping of a distributed identity is mapped to a RACF user ID, which is then associated with a authorization ID, and then linked to a DB2 role based in the trusted context definition. The distributed identities can be switched on the same connection without be authenticated.

Note: At the time of the writing, the length of the user ID in DB2 was limited to 8 character bytes. However, this restriction has been lifted by APAR PM31429. Because the distributed identity is not a defined RACF user ID, DB2 will not impose any length restrictions on the user ID (which might contain a distinguished name in the x.500 format).

Refer to the “Example of establishing an explicit trusted connection and switching the user” section in *DB2 Version 9. for Linux, UNIX and Window*, SC27-2443-01.

3.3.1 DB2 10 for z/OS Trusted Context

A *trusted connection* is a database connection that is established when the connection attributes match those of a *trusted context* that is defined at the server. A trusted connection can be established locally or at a remote location.

DB2 10 for z/OS Identity Propagation works only in a trusted context. The end user’s identity consists of a distinguished name and registry name. Using a trusted connection, DB2 for z/OS detects the registry name and allows the distributed identity to be passed onto RACF to be mapped.

Note: Refer to chapter 8, “Managing Access through trusted context,” in the *DB2 10 for z/OS Administration Guide*, SC19-2968-01.

Creating a trusted context

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

Using trusted connections provides the ability for an external entity with a distinguished name and realm to use an established trusted connection to the database server, without being reauthenticated at the database server. DB2 then passes on the distributed identity to RACF to be mapped against a RACF user ID.

A trusted connection can be established from a remote application after DB2 matches the connection attributes setup in the trusted context. Use the CREATE TRUSTED CONTEXT statement to define the trusted context in DB2. Refer to *DB2 DB2 10 for z/OS SQL Reference*, SC19-2983-01, for more information about the create context statement.

3.3.2 RACMAP

RACF Class IDIDMAP must be activated before you issue the RACMAP command. RACMAP commands are used to map the distributed identity to the RACF user ID. The distributed identity DN is stored in the USERDIDFILTER and realm is the REGISTRY name.

The registry name is not verified by RACF, however, it must be supplied by the distributed application, otherwise DB2 will not recognize the distributed identity and will be treated as a normal RACF user ID, which then abends on the client with:

```
-4499 An error occurred during a deferred connect reset and the connection has  
been terminated
```

Registry name (realm)

The registry name contents defined in RACMAP are not verified by RACF or DB2. However, a registry name must be supplied so that DB2 allows the distributed identity to be forwarded to RACF and mapped. Otherwise, it is treated as a normal user ID and abends with:

```
-4499 An error occurred during a deferred connect reset and the connection has  
been terminated
```

The registry name on RACMAP must be captured as an asterisk (*), which means that we allow any registry or realm to connect.

USERDIDFILTER

The Userdidfilter field contains the distinguished name. Currently, DB2 10 for z/OS only allows for an 8-character distinguished name. The userdidfilter is the single matching component of the distributed identity. If the matching distinguished name and userdidfilter do not exist, it is rejected.

Refer to Chapter 6, “Using SMF audit information to report on z/OS Identity Propagation” on page 55, for a more detailed explanation of the RACMAP command and what each parameter means. Also, refer to the example used with DB2 in Chapter 9, “Identity propagation with DB2 for z/OS” on page 113.

3.3.3 DB2 role in RACF

A *privilege* enables the user of an ID to execute certain SQL statements or to access the objects of another user. A *role* groups the privileges together so that they can be simultaneously granted to and revoked from multiple users.

A role is a database entity that is created in DB2. It is defined through the SQL CREATE ROLE statement and is associated with an authorization ID in a trusted context definition. A role cannot be used outside of a trusted context unless the user in a role grants privileges to an ID.

DB2 roles are possible when using the RACF/DB2 exit. Although not a defined entity in RACF, it can be associated with a RACF user ID using the WHEN CRITERIA option in the RACF PERMIT command. Before using DB2 roles with the RACF access control module, the security administrator must define DB2 resources profiles.

Note: For more details about the RACF/DB2 exit and the RACF setup, refer to *DB2 10 for z/OS RACF Access Control Module Guide*, SC19-2982-01.

3.3.4 SMF reporting

The RACF and DB2 SMF reporting combined can provide valuable auditing reports now, where the distributed identity can be reported on z/OS SMF data.

Note: Refer to Chapter 4, “RACMAP function” on page 39.

RACF SMF Type 80

RACF SMF Type 80 records show you the mapping that was performed. Type 80 is now able to display the distributed identity in the audit trail. This means that we are able to identify the individual from a remote system, accessing the backend z/OS server.

Using the RACF/DB2 exit provides you with additional information regarding which DB2 object profile access was successful. Not using the RACF/DB2 exit, and reporting on RACF SMF Type 80, you are still able to see the mapping of the distributed identity. See Chapter 9, “Identity propagation with DB2 for z/OS” on page 113, for an example of this.

Note: Refer to Chapter 4, “RACMAP function” on page 39.

DB2 audit trace

We have used DB2 10 for z/OS Audit Policy to produce the SMF data used in our examples. Audit policies can be created by adding a record to the new SYSIBM.SYSAUDITPOLICIES catalog table. After an audit policy has been added, you can use the -START TRACE(AUDIT) AUDTPLCY(policyname).

command to start the trace for the specified policyname.

Refer to Chapter 11, “Auditing access to DB2,” in the *DB2 10 Administration Guide*, SC19-2968-01, for more information about how to create and activate DB2 audit policy.

This audit policy feature simplifies the process, and by using the VERIFY audit category, the following can be reported on:

- ▶ IFCID 55 - set current sqlid
- ▶ IFCID 83 - identify request
- ▶ IFCID 87 - signon request
- ▶ IFCID 169 - distributed authid translation
- ▶ IFCID 269 - trusted connection established/reused

Using SMF reporting tools, the fields within these IFCIDs should be sufficient to produce a report on the distributed identities, which user ID it was mapped to, the trusted context used, the role, the objects accessed, and whether the access was successful.

Chapter 9, “Identity propagation with DB2 for z/OS” on page 113, provides a test with examples.

Dynamic Statement Cache

Having dynamic statement cache activated, and with the AUTHID and SQL statement being reused, DB2 would not fully authenticate each time. For example, when the RACF/DB2 exit is active, you will not see a RACF SMF type 80 being produced each time that the same AUTHID is executing the same SQL.

With identity propagation, we expect different distributed identities, which could get mapped to the same AUTHID. This can be reported on with each RACINIT produced by RACF SMF type 80.

If, however, you flush the dynamic statement cache, then RACF performs the authentication against the table profile, and you get a new RACF SMF type 80 produced, showing the access READ to the SELECT profile.

The same applies when not using the RACF/DB2 exit. DB2 does not re-authenticate if the SQL statement and AUTHID are located in the dynamic statement cache.

3.3.5 Distinguished name not matching

If there is no match for a distinguished name in the IDIDMAP class profiles, then the authorization fails (Figure 3-7).

```
ICH408I USER(STC      ) GROUP(TSO      ) NAME(STARTED TASK      )
  DISTRIBUTED IDENTITY IS NOT DEFINED:
  TestUser
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.
ICH408I USER(TESTUSER) GROUP(          ) NAME(???                )
  LOGON/JOB INITIATION - USER AT TERMINAL      NOT RACF-DEFINED
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.
DSNL030I  -DBOD DSNLTSEC.30 DDF PROCESSING FAILURE 777
FOR
LUWID=G90C0594.H1FD.C747566B6C9D
  REASON=00F30089,THREAD-INFO=ROGERIO:thumbiran:rogerio:db2jcc_applicat
  REASON=00F30089,THREAD-INFO=ROGERIO:thumbiran:rogerio:db2jcc_applicat
  ion:*:*:DEFREAD
```

Figure 3-7 Distributed identity is not defined

In Figure 3-7, we test with distributed identity TestUser, which does not have a RACMAP definition. RACF then displays DISTRIBUTED IDENTITY NOT DEFINED. Again, it shows the user ID TESTUSER, *not* RACF-DEFINED.

Also, at the client end, using our JAVA application test, we get the following error:

```
-4499 An error occurred during a deferred connect reset and the connection has
been terminated
```



RACMAP function

This chapter discusses the new RACF RACMAP command, which was introduced with z/OS V1R11, and the association between distributed identity filters and RACMAP.

Significant improvements have been made to RACF to support distributed identity propagation, and the RACF RACMAP command is one of the important keys in this scenario. Beginning with z/OS V1R11, RACF accepts information about the identities of distributed users from authorized applications that issue the RACROUTE REQUEST=VERIFY.

This chapter describes the following updates to the RACMAP and distributed identity filters:

- ▶ Brief overview of distributed identity filters and how RACF is related to them
- ▶ RACMAP concepts, usage, and invocation
- ▶ RACF commands for RACMAP
- ▶ RACF updates to support RACMAP

4.1 Distributed identity filters

Currently, many transactions that execute on z/OS subsystems from the internet are initiated by users who authenticate their identities on distributed application servers. After such distributed application servers send a transaction to a z/OS subsystem, this transaction might be associated with the identity of the distributed application user or associated with a shared RACF user ID that was assigned by the z/OS subsystem.

The issue addressed by RACMAP is that auditing this user activity on z/OS needs the RACF user ID associated with a z/OS subsystem transaction *and* the user identity that was presented when the user initially accessed the distributed application server.

The RACMAP command addresses this issue by allowing both user identities to be recorded in the SMF records, providing accurate auditing.

4.1.1 What a distributed identify filter is

A distributed identify filter consists of one or more components of a distributed user's name and the name of the registry where the user ID is defined, or we might simply say that distributed identity filter is the mapping association between a RACF user ID and one or more distributed user identities.

4.2 RACMAP command overview

The RACMAP command can be used to create, delete, and list a distributed identity filter. When you use the RACMAP command to define a distributed identity filter, RACF creates a general resource profile in the IDIDMAP class. RACF exploits the use of the distributed identity filter to determine the RACF user ID of a user who tries to access the system using a distributed identity.

The RACMAP command has the following functions:

- ▶ MAP: Creates a distributed identity filter
- ▶ LISTMAP: Lists information about a distributed identity filter
- ▶ DELMAP: Deletes a distributed identity filter
- ▶ QUERY: Finds the matching RACF user ID associated with a distributed identity filter

4.3 Authorization required to use the RACMAP command

To issue the RACMAP command, you must have System SPECIAL authority or sufficient authority to the IRR.IDIDMAP.*function* resource in the FACILITY class, where *function* is MAP, DELMAP, LISTMAP, or QUERY.

Table 4-1 lists the authority required to use the RACMAP command.

Table 4-1 Authority required for the RACMAP command

Access level	Purpose
READ	Create, delete, or list a distributed identity filter for your own RACF user ID.
UPDATE	Create, delete, or list a distributed identity filter for another RACF user ID.

4.4 RACMAP command usage and invocation

RACMAP can be entered as the RACF TSO command. The syntax of the RACMAP TSO command is the same as the general syntax of other TSO commands. For example, a comma or one or more blanks are valid delimiters for use between operands. You can also enter commands in the background by using a batch job.

RACMAP	
	[ID(<i>userid</i>)]
	MAP USERDIDFILTER(NAME('distributed-identity-user-name' '*')) REGISTRY(NAME('distributed-identity-registry-name' '*')) [WITHLABEL('label-name')]
	DELMAP [(LABEL('label-name'))]
	LISTMAP [(LABEL('label-name'))]
	QUERY USERDIDFILTER(NAME('distributed-identity-user-name')) REGISTRY(NAME('distributed-identity-registry-name'))

Figure 4-1 RACMAP command syntax

Figure 4-1 shows the syntax of the RACMAP command, where the RACMAP parameters are:

- ▶ ID(*userid*) specifies the RACF user ID mapped by the distributed identity filter. The user ID must already be defined to the RACF database.
- ▶ MAP specifies the MAP function of the RACMAP command. Use the MAP function to create a distributed identity filter that maps a user's distributed identity to a RACF user ID.

Note: The MAP function creates a profile in the IDIDMAP class for each filter created.

When using the MAP function, you must specify values for the required USERDIDFILTER and REGISTRY parameters:

- USERDIDFILTER(NAME('distributed-identity-user-name'|*))

This specifies the significant portion of the distributed-identity user name.

The maximum length for a user name is 246 bytes.

In general, the user name can contain blank and mixed-case characters, however, the value cannot be specified as a hexadecimal character string.

Format of the user name value: Specify the user name value in any of the following three formats:

- As a single asterisk (*) to indicate that any user name matches this filter:
 USERDIDFILTER(NAME('*'))
- As a simple character string, such as a user ID defined in a non-LDAP registry:
 USERDIDFILTER(NAME('DENICE'))

- As a character string that represents an X.500 distinguished name. A distinguished name (DN) consists of one or more relative distinguished names (RDNs). Each RDN consists of an attribute type and attribute value, separated by an equals sign (=). RDNs are separated by a comma (,):

```
USERDIDFILTER(NAME('UID=GUSKI,OU=Tools,O=IBM,C=US'))
```

Note: If you specify mixed-case characters in the user name as a DN, the RACMAP command translates the attribute field type to uppercase and preserves the mixed-case characters of the field value.

- REGISTRY(NAME('distributed-identity-registry-name'|**))

This specifies the registry that contains the distributed-identity user name. You can specify a single asterisk (*) as the registry name to indicate that any distributed-identity registry name matches this filter.

The maximum length for a registry name is 255 bytes. For example, using the REGISTRY name:

```
REGISTRY(NAME('ldaps://us.richradioham.com'))
```

WITHLABEL('label-name') specifies the label assigned to this distributed identity filter. This parameter is not required. However, if specified, the label must be unique to the RACF user ID associated with this filter.

In case where WITHLABEL is not specified, RACF automatically generates a label for the filter in the form of LABELnnnnnnnn, where nnnnnnnn is the first integer value that generates a unique label name.

The maximum length for a label-name is 32 characters.

- ▶ DELMAP specifies the DELMAP function of the RACMAP command. It must be used to delete a distributed identity filter for the specified RACF user ID. The DELMAP function also deletes the profiles from the IDIDMAP class.

If the specified RACF user ID is associated with more than one filter, you must also specify the label name, For example:

```
RACMAP ID(userid) DELMAP LABEL('label-name')
```

- ▶ LISTMAP specifies the LISTMAP function of the RACMAP command. Use it to list information about a distributed identity filter for the specified RACF user ID.

If the specified RACF user ID is associated with more than one filter, you must also specify the label name, for example:

```
RACMAP ID(userid) LISTMAP LABEL('label-name')
```

You can also omit LABEL to list all filters associated with the specified RACF user ID, for example:

```
RACMAP ID(userid) LISTMAP
```

- ▶ QUERY specifies the QUERY function of the RACMAP command. Use the QUERY function to find the matching RACF user ID that is associated with a distributed identity filter.

Note: When you specify the QUERY function, you must specify both USERDIDFILTER and REGISTRY.

4.5 Activating the RACMAP updates

To activate your changes in the IDIDMAP class, activate and RACLIST the IDIDMAP class. After a distributed identity filter is created for the first time, issue the command to activate and RACLIST your IDIDMAP class:

```
SETROPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)
```

Where:

- ▶ CLASSACT specifies that you want the IDIDMAP class to be in effect.
- ▶ RACLIST specifies that you want to activate the sharing of in-storage profiles for this class.

If the IDIDMAP class is already active and RACLISTed, you just need to refresh the class to validate your changes:

```
SETROPTS RACLIST(IDIDMAP) REFRESH
```

4.6 RACMAP profiles in the IDIDMAP class

Each distributed identity filter is stored in a general resource profile in the IDIDMAP class. Every time that the RACMAP MAP command is issued, RACF creates a general resource profile in the IDIDMAP class, and when you use the RACMAP DELMAP command, RACF deletes the IDIDMAP profile.

If you specify multiple filters for a particular user name with each filter specifying a different registry, your IDIDMAP profile might contain multiple filters.

The name of an IDIDMAP profile is the user name of the filter that you specify in the USERDIDFILTER value and is encoded as UTF-8 data.

All RACF profiles from the IDIDMAP class have an owner who is the user ID of the RACMAP MAP command issuer. Although each IDIDMAP profile has an owner, the profile owner has no authority over the IDIDMAP profile. RACF does not use profile owner information for any purpose. Also, the profile owner of an IDIDMAP profile cannot be changed.

The profile owner can be found in the output of the RACF database unload (IRRDBU00) utility and also through the use of RL LIST IDIDMAP * command. However, the RLIST command must not be used to manage RACMAP information.

Note: Use the RACMAP command to administer distributed identity filters. Do not use the RDEFINE, RALTER, REDELETE, or RLIST RACF commands to administer the IDIDMAP profiles.

Considerations about UTF-8 data values

Most RACF profile data is stored as EBCDIC data. However, the user and registry names that were specified in the USERDIDFILTER and REGISTRY parameters from the RACMAP command are not stored in IDIDMAP as EBCDIC data. Instead, they are encoded as UTF-8 data and stored in hexadecimal format. Therefore, they are compatible with the typical processing of X.500 distinguished names.

Restrictions for UTF-8 data values

Because the IDIDMAP profile name (derived from the user name) and the registry name are encoded as UTF-8 data, the following restrictions apply to UTF-8:

- ▶ When using the RACMAP command to define user and registry names that contain multibyte characters, if the resulting UTF-8 values exceeds 246 bytes for a user name or 255 bytes for a registry name, the RACMAP MAP command fails with the message IRRW2131.
- ▶ When using the SEARCH command, you cannot use the FILTER or MASK option to limit your search results based on the names of the IDIDMAP profiles.

4.7 Updating a distributed identity filter

If changes are needed to a distributed identity filter, you need to delete the current filter and define a new one using the RACMAP MAP command. A distributed identity filter cannot be updated or modified without using the RACMAP DELMAP command and then issuing a RACMAP MAP command.

4.7.1 Steps for updating a distributed identity filter

In the example shown in Figure 4-2, a distributed identity mapping is defined.

```
Mapping information for user ROGER:

Label: REGISTRY01
Distributed Identity User Name Filter:
  >IBMUSER001<
Registry Name:
  >ldaps://ibm.com.us<
```

Figure 4-2 Mapping information from user Roger

As an example, we want to change the label from REGISTRY01 to REGISTRY02 of the user identity filter ROGER. Because there is no RACMAP command that can be used to update or modify a current distributed identity filter, we need to delete the current identity filter and create a new one specifying the label REGISTRY02:

1. Issue the RACMAP command with the DELMAP function:

```
RACMAP ID(ROGER) DELMAP
```

2. Create the new RACMAP identity filter using the MAP function:

```
RACMAP ID(ROGER) MAP USERDIDFILTER(NAME('IBMUSER001'))
REGISTRY(NAME('ldaps://ibm.com.us')) WITHLABEL('REGISTRY02')
```

3. Issue the RACF REFRESH command to refresh the IDIDMAP class:
SETROPTS RACLIST(IDIDMAP) REFRESH
4. List the new RACMAP identity filter using the LISTMAP function (Example 4-1).

Example 4-1 RACMAP LISTMAP command

```
RACMAP ID(ROGER) LISTMAP
```

Mapping information for user ROGER:

```
Label: REGISTRY02
Distributed Identity User Name Filter:
  >IBMUSER001<
Registry Name:
  >ldaps://ibm.com.us<
```

4.8 User profiles and RACMAP command

In addition to creating an IDIDMAP profile, the RACMAP command also updates the RACF user profile. Therefore, when you delete the filter, RACMAP deletes the IDIDMAP profile containing the filter and updates the user profile, removing the mapping association.

4.8.1 Deleting a RACF user ID associated with identity filters

If you need to delete a RACF user profile from the RACF database and this same user profile has a distributed identity filter, you will not be able to delete the RACF user profile while this association is still valid. That is, you cannot delete a RACF user profile that is associated with a distributed identity filter.

First, you need to delete the mapping association from the user profile by issuing the RACMAP DELMAP command to delete the filter. After that you will be able to delete the user profile with the DELUSER command.

The message in Example 4-2 is issued if you try to delete a user profile with a remaining distributed identity filter association.

Example 4-2 RACF warning message

```
ICH04018I userid cannot be deleted. Distributed identity mapping profiles are
associated with this user.
```

4.8.2 Performance consideration when deleting a distributed identity filter

As stated earlier, if changes are needed to a filter, you should delete it and create a new one with the new parameters, because modifying a distributed identity filter is not possible. However, when the RACMAP DELMAP command is issued specifying both filter label and a user ID for which no user profile exists, RACF searches all profiles in the IDIDMAP class to locate and delete all matching filters. This search might take an extended period of time.

4.8.3 RACF remove ID utility IRRRID00 update

The RACF IRRRID00 utility was updated to create RACMAP DELMAP commands to clean up IDIDMAP profiles and to locate residual user IDs in IDIDMAP profiles. IRRRID00 helps you keep your RACF database current.

4.9 Default RACMAP filter protection

You can create a default RACMAP filter by specifying an asterisk (*) as the distributed identity user name and also as the registry name. Example 4-3 shows how you can map all distributed identity names that are unmapped by more specific filters by defining a default RACMAP filter, which we called NORACMAP.

Example 4-3 RACF example of defining a default RACMAP protection

```
RACMAP ID(NORACMAP) MAP USERDIDFILTER(NAME('*')) REGISTRY(NAME('*'))
```

After the default RACMAP is defined, remember to assign the RACF user ID with both restricted and protected RACF attributes. This way, your default RACMAP user ID cannot be used to access protected resources because it is not specifically authorized to access these, nor can it be used to log on to the system.

Example 4-4 shows an example by specifying restricted and protected attributes.

Example 4-4 RACF command specifying restricted and protected attributes

```
ALU NORACMAP RESTRICTED NOPASSWORD NOPHRASE
```

Where:

- ▶ RESTRICTED specifies that global access checking is bypassed when resource access checking is performed for the user, and neither ID(*) on the access list nor the UACC will allow access.
- ▶ NOPASSWORD specifies that the user cannot use a password for authentication.
- ▶ NOPHRASE specifies that the user cannot use a password phrase for authentication.

4.10 RRSF consideration for RACMAP use

As discussed earlier, the RACMAP command updates profiles in the user and IDIDMAP classes. Updates to profiles in these classes with the RACMAP command are eligible for automatic direction of application updates.

To ensure that RACF database updates are propagated in a consistent manner across the IDIDMAP and user classes, define an RRSFDATA resource called AUTODIRECT.*target-node*.IDIDMAP.APPL.

4.11 Changes required to PARMLIB to support identity filter

The following member IKJTSOxx from your PARMLIB must be updated to support the RACMAP function. This member is used at IPL time to define, among other things, the

authorized command list. During the IPL, the IKJTSOxx member is read. You must edit this member and include the RACMAP command into the AUTHCMD list so that the RACMAP is automatically activated at each IPL time.

If you do not update your IKJTSOxx at IPL time, you need to edit it and add the RACMAP command and then issue the dynamic PARMLIB command using the UPDATE operand and specify the suffix of your member (xx) that you have edited. You must have UPDATE authority to the proper RACF resource class to issue the dynamic PARMLIB command.

Otherwise, if the IKJTSOxx is not updated with the RACMAP command, you will receive the abend message shown in Figure 4-3 after you try to execute the RACMAP function.

```
Abend 684000 hex occurred processing command 'RACMAP '.  
ISPD210 CMD abended - 'RACMAP' terminated abnormally.
```

Figure 4-3 RACMAP abend message

4.12 New RACMAP messages

The new messages related to the RACMAP command are:

IRRW201: You are not authorized to issue the RACMAP command.

Explanation: You do not have the required authority to issue the RACMAP command.

IRRW202: The user ID specified is not defined to RACF.

Explanation: The user ID specified on the ID keyword of the RACMAP command could not be found in the RACF database.

IRRW203: Unexpected ICHEINTY error encountered during command processing. ICHEINTY RC = x'*retcode*', ICHEINTY RSN = x'*rsncode*'.

Explanation: During command processing RACMAP issued an ICHEINTY and received a return code and reason code that were not expected.

IRRW204: No information was found for user *userid*.

Explanation: RACMAP was unable to find distributed identity information for the user ID that is indicated in the message.

IRRW205: Additional information is required to identify the identity mapping.

Explanation: RACMAP found more than one distributed identity mapping for this user. The information required to uniquely identify the mapping was not provided.

IRRW206: No matching identity mapping was found for this user.

Explanation: RACMAP could not find a mapping profile for the specified user that matches the label provided.

IRRW207: Unexpected RACROUTE REQUEST=*request-type* error encountered during command processing. SAF RC = x'*retcode*', RACF RC = x'*retcode*', RACF RSN = x'*rsncode*'.

Explanation: During command processing RACMAP issued a RACROUTE request of the specified type but received an unexpected return code and reason code.

IRRW208: The label *label-name* is already in use.

Explanation: You attempted to associate a user ID with a mapping profile and assign *label-name* to that association. This label is already in use for this user ID.

IRRW209: This filter already exists. It cannot be added.

Explanation: You attempted to add a filter that already exists in a mapping profile in the IDIDMAP class. Filters must be unique.

IRRW210: RACLISTed profiles for the IDIDMAP class will not reflect changes until a SETROPTS RACLIST REFRESH is issued.

Explanation: RACF uses copies of the IDIDMAP profiles that exist in a dataspace. Updates made to IDIDMAP profiles become effective only when the SETROPTS command is issued with the REFRESH operand.

IRRW211: Registry information is required.

Explanation: You specified a MAP request, but did not specify the registry name.

IRRW212: Distributed user identity information is required.

Explanation: You specified a MAP request, but did not specify the distributed user identity information.

IRRW213: An error occurred while converting the data for the keyword-name keyword from EBCDIC to UTF-8.

Explanation: While converting the value specified for the USERDIDFILTER or REGISTRY keyword from EBCDIC format to UTF-8 format, the UNICODE services returned an unexpected return code. The value specified might contain multibyte characters causing the value to exceed the byte limit.

IRRW214I: The *KeyWord-Name* keyword is ignored when specified with the *Function-Name* function.

Explanation: You specified a keyword that is not needed by the function.

IRRW215I: No user ID found associated with the specified USERDIDFILTER and REGISTRY name.

Explanation: The information that you provided with the UserDIDFilter and registry names is not associated with any RACF user ID.

IRRW216I: Unexpected *Callable-Service-Name* callable service error encountered during command processing. SAF RC = x'RetCode', RACF RC = x'RetCode', RACF RSN = x'RsnCode'.

Explanation: During command processing, RACMAP issued a call to this callable service and received a return code and reason code that were not expected.

ICH04018I: *userid* cannot be deleted. Distributed identity mapping profiles are associated with this user.

Explanation: The indicated user ID has not been deleted from the RACF database because the user profile indicates that distributed identity mapping profiles still exist for the user in the IDIDMAP class.

ICH408I: DISTRIBUTED IDENTITY IS NOT DEFINED: *distributed-identity-information*.

Explanation: A user attempted to access a server using a distributed identity that is not associated with a RACF user ID. RACF cannot determine a user ID for this user.



Filter management

This chapter discusses aspects of filter management and how RACF with the RACMAP command matches the specified distributed filter, and it provides distributed identity filter examples.

5.1 How RACF matches the filter value

When a user authenticates from a distributed application server and takes an action that causes a supported transaction to be sent to the z/OS system, RACF receives the user's distributed identity and registry names as character string of UTF-8 data. When the IDIDMAP class is active and RACLISTed, RACF uses the UTF-8 data to search IDIDMAP profiles for the distributed identity filter that contains the name values that best match the data. When the best matching filter is found, RACF then assigns a RACF user ID.

5.2 Details about searching for a filter that matches a user's DN

You can specify user and registry name values in the distributed identity filter to map a RACF user ID using a *one-to-one* match or a *many-to-one* match. That is, you can define a filter that assigns a RACF user ID to only one distributed user, or you can define a filter that assigns the same RACF user ID to multiple distributed users.

5.2.1 One-to-one match

A one-to-one match is a filter that maps a RACF user ID to only one distributed user that contains a registry name value and contains a user name value that is specified in one of the following ways:

- ▶ As a user ID or user name defined in a non-LDAP registry

When you specify the user name in this way, both the distributed user's registry and user name must exactly match the registry and user name values in the filter.

An example of defining a filter for a non-LDAP user name is when the user DENICE authenticates her user identity at her distributed application server and takes an action that causes a transaction to be sent to the z/OS system, and then RACF is passed the following distributed user and registry names as character strings of UTF-8 data:

- DENICE
- Registry01

When RACF uses these data values to search the IDIDMAP profiles for a matching filter, RACF finds a match to the filter labeled "Filter for Denice from Registry01" and assigns the DENICE user ID. The filter has been defined as shown in Example 5-1.

Example 5-1 Filter for Denice

```
RACMAP ID(DENICE) MAP USERDIDFILTER(NAME('DENICE'))  
REGISTRY(NAME('Registry01')) WITHLABEL('Filter for Denice from Registry01')
```

The transaction executes with authority of the DENICE user ID. Any audit records that are written for this transaction contain both the RACF user ID and the original distributed user and registry names that were passed to RACF when the transaction was sent.

- ▶ As an X.500 DN that includes all RDNs necessary to uniquely identify the distributed user. Depending on the particular LDAP registry, the DN might include the UID or CN components to uniquely identify the user.

When you specify the user name in this way, the distributed user's registry must exactly match the registry name value in the filter, and the distributed user's name must exactly match all RDNs specified in the user name value in the filter.

As an example of defining a filter for a full X.500 DN, when Bob Cook authenticates his LDAP user identity at his distributed application server and takes an action that causes a transaction to be sent to the z/OS system, RACF is passed the following distributed user and registry names as character strings of UTF-8:

- UID=BobC,CN=Bob Cook,OU=Accounting,O=BobsMart,C=US
- ldaps://us.bobsmarturl.com

When RACF uses these data values to search the IDIDMAP profiles for a matching filter, RACF finds an exact match to the filter labeled “Accounting boss” and assigns the RLCOOK user ID. The filter has been defined as shown in Example 5-2.

Example 5-2 Filter for accounting boss

```
RACMAP ID(RLCOOK) MAP USERDIDFILTER(NAME('UID=BobC,CN=Bob
Cook,OU=Accounting,O=BobsMart,C=US'))
REGISTRY(NAME('ldaps://us.bobsmarturl.com')) WITHLABEL('Accounting boss')
```

The transaction executes with the authority of the RLCOOK user ID. Any audit records that are written for this transaction contain both the RACF user ID and the original distributed user and registry name that were passed to RACF when the transaction was sent.

5.2.2 Many-to-one match

A many-to-one match is a filter that maps the same RACF user ID to multiple distributed users that are specified in any of the following ways:

- ▶ The registry name value is specified as a single asterisk (*) to indicate that any registry name matches the registry portion of the filter.
 - When you specify the registry name in this way and you specify a user name value, the distributed user’s name must exactly match the user name value in the user portion of the filter.
 - When you specify each of the user and registry name values as an asterisk, any distributed user’s name from any registry matches the filter. This type of filter is called a default RACMAP filter.
- ▶ The user name is specified in one of the following ways:
 - As an X.500 DN that includes selected RDNs that are common to multiple distributed users. Depending on the particular LDAP registry, the specified DN would likely omit the UID or CN components.

When you specify the user name in this way and you also specify a registry name value, the distributed user’s registry must exactly match the registry name value in the filter, and the distributed user’s name must match one or more RDNs in the user name value of the filter.

An example of defining a filter using selected RDNs, is when an accounting office worker named Lila Jones authenticates her LDAP user identity at the distributed application server and takes an action that causes a transaction to be sent to the z/OS system. RACF is passed the following distributed user and registry names as character strings of UTF-8 data:

- UID=LJones,CN=Lila Jones,OU=Accounting,O=BobsMart,C=US
- ldaps://us.bobsmarturl.com

When RACF uses these data values to search the IDIDMAP profiles for a matching filter, RACF finds no match. When no match is found, RACF removes the most specific portion of the user name, the first RDN of the DN (UID=LJones), and performs a

second search of the IDIDMAP profiles. When no matching filter is found, RACF removes the first two RDNs (UID=LJones,CN=Lila Jones) and performs a third search of the IDIDMAP profiles. This time, RACF finds a match to the filter labeled “Accounting office workers” and assigns the ACCTUSER user ID.

The transaction that Lila initiated executes with the authority of the ACCTUSER user ID. Any audit records that are written for this transaction contain the ACCTUSER user ID and the original distributed user name (including all RDNs) and the registry name for user Lila Jones, which were first passed to RACF when the transaction was sent.

- As a single asterisk (*) to indicate that any user name matches the user portion of the filter.

When you specify the user name as an asterisk and specify a registry name value, only the distributed user’s registry must match the registry name value in the filter. Any distributed user from the specified registry matches the filter.

When you specify each of the user and registry name values as an asterisk, any distributed user’s name from any registry matches the filter. This type of filter is called a default RACMAP filter.

5.2.3 Summary details about searching for a filter that matches a user’s DN

When RACF searches for the distributed identity filter that best matches a user’s DN, RACF attempts to match the user’s registry name and exactly match all RDNs of the user’s DN.

If a matching filter is found, RACF assigns the user ID specified by the filter.

If no matching filter is found, RACF ignores the most specific or first RDN of the user’s DN, for example, UID, and performs a second search to locate a less restrictive filter. If a less restrictive filter is found, RACF assigns the user ID specified by the filter.

If no matching filter is found, RACF ignores the first two RDNs, for example, UID and CN, and performs a third search. If no matching filter is found, RACF iteratively ignores each subsequent RDN, searching for a less restrictive filter, until the last RDN is used.

If no matching filter is found, RACF searches for a filter that matches the user’s registry name and contains an asterisk as the user name. If a matching filter is found, RACF assigns the user ID specified by the filter.

If no matching filter is found, RACF searches for the default RACMAP filter. If the default filter is defined, RACF assigns the user ID that it specifies. If no default filter is found, RACF assigns no user ID.

5.3 Examples

The following examples illustrate possible situations in which the security administrator is requested to create or manage a distributed identity filter:

- ▶ Example 1: The security administrator wants to add a distributed identity filter that specifies the distributed user’s name using all RDNs of the user’s X.500 distinguished name:

```
RACMAP ID(RLCOOK) MAP USERDIDFILTER(NAME('UID=BobC,CN=Bob
Cook,OU=Accounting,O=BobsMart,C=US'))
REGISTRY(NAME('ldaps://us.bobsmartur1.com')) WITHLABEL('Accounting boss')
```

- ▶ Example 2: The security administrator wants to add a distributed identity filter that specifies the distributed user's name using selected RDNs of the user's X.500 distinguished name:


```
RACMAP ID(ACCTUSER) MAP USERDIDFILTER(NAME('OU=Accounting,O=BobsMart,C=US'))
REGISTRY(NAME('ldaps://us.bobsmarturl.com')) WITHLABEL('Accounting office
workers')
```

- ▶ Example 3: The security administrator wants to add a distributed identity filter that specifies the distributed user's name as a non-LDAP user name:

```
RACMAP ID(DENICE) MAP USERDIDFILTER(NAME('DENICE'))
REGISTRY(NAME('Registry01')) WITHLABEL('Filter for Denice from Registry01')
```

- ▶ Example 4: The security administrator wants to delete the distributed identity filter labeled "Filter for Denice from Registry01" for the RACF user ID DENICE:

```
RACMAP ID(DENICE) DELMAP(LABEL('Filter for Denice from Registry01'))
```

Using SMF audit information to report on z/OS Identity Propagation

This chapter describes how z/OS System Management Facility (SMF) can be used to provide audit information obtained about identity propagation. That is, it allows an auditor to map a z/OS Security Server user ID (more commonly referred to as a RACF user ID) to an entity who entered the z/OS system from a distributed system.

It explains how RACF reacts when encountering instances of identity propagation, how RACF communicates audit information to SMF, what products we can use to report from that data, and examples of those reports.

Data contained in the SMF data records is easily reported upon. A number of products are available that can do this. This chapter identifies some of these products and gives examples of their output showcasing the data relevant to identity propagation.

6.1 Actions within RACF

SMF will contain data records written to it by the z/OS Security Server when it encounters certain events. What we are interested in is what outcomes occur for these actions relevant to identity propagation when an event occurs within RACF. So, the following discussion is not a full explanation of a RACF action, but describes those portions relevant to the topic of this publication.

6.1.1 An RACF event relating to issuing a RACMAP command

When a RACMAP command, which creates or deletes a IDIDMAP profile, occurs an event code 87 occurs within RACF that, if permitted, writes a SMF type 80 record. Table 6-3 on page 58 addresses permissions to write SMF records. The significant information that is contained in this particular SMF record is:

- ▶ RACF user ID
- ▶ Value of the user ID filter name from the USDERDIDFILTER keyword of RACMAP
- ▶ Value of the registry name from the REGISTRY keyword of RACMAP

This gives us confidence that when we define a mapping of a distributed user to a RACF user ID, we capture this audit information in a SMF record along with who performed this RACF command.

6.1.2 RACF events when calling RACF to verify a distributed identity

z/OS subsystems (for example, CICS) can make an enquiry to RACF to provide it with a RACF user ID based on the supplied distributed identity values. This call is a RACROUTE, and the request made is VERIFY or VERIFYX. RACF can issue an event code 1 when the distributed identity cannot be found to map to a RACF user ID.

An event code 67 occurs when authorized applications, such as web servers, invoke a initACEE callable service (IRRSIA00) to request either to create a security context created or have the distributed identity queried and the status returned.

Table 6-1 describes event codes upon failure to map a distributed identity.

Table 6-1 Event Codes upon failure to map a distributed identity

Event code	Event qualifier	Event qualifier number	Event description
1	DIDNOTDF	39	No RACF user ID found for distributed identity
67	DIDNOTDF	10	No RACF user ID found for distributed identity

Note: Each unique event code has a corresponding event qualifier. The event qualifier has a value that indicates whether the event succeeded or failed.

6.1.3 Settings within RACF to ensure identity propagation is captured

Having seen what happens with events in RACF in relation to identity propagation, we need the correct RACF settings in place to permit data records to be written to SMF. If more than one setting applies to a particular activity within RACF, only one SMF data record will result, but it will contain information within it to indicate which settings were active when it was written.

The following points relate to the auditing options of the SETROPTS (set RACF options) command:

► **AUDIT**

This specifies the names of the classes for which you want RACF to perform auditing. When the class specified is USER, RACF logs all password and password phrase changes made by RACROUTE REQUEST=VERIFY. (RACF will add the classes you that specify to those already specified for auditing.)

If you specify an asterisk (*), logging occurs for all classes.

You must have the AUDITOR attribute to enter the AUDIT operand.

► **CMDVIOL**

This specifies that RACF is to log violations detected by RACF commands during RACF command processing. A violation might occur because a user is not authorized to modify a particular profile or is not authorized to enter a particular operand on a command.

You must have the AUDITOR attribute to specify these options.

► **SAUDIT**

This specifies that RACF is to log RACF commands issued by users who either had the SPECIAL attribute or who gained authority to issue the command through the group-SPECIAL attribute. You must have the AUDITOR attribute to specify these operands.

RACF also provides the ability to allow audit information to be captured when an activity occurs at the profile level for a user. The auditor can selectively audit the actions of a specific user using the UAUDIT operand on the RACF RALTER command.

6.2 SMF changes to support distributed identities

This chapter shows that through a variety of RACF events, written SMF data records will contain distributed identity information. SMF Type 80 records hold a range of auditing information and thus provide a valuable resource for subsequent data analysis and reporting. This latter function occurs after the SMF data has been unloaded from the active SMF data set.

6.2.1 SMF records

IBM Security zSecure can report directly from an active SMF data set. This is shown in Figure 6-1 as a sequence of Consul Auditing and Reporting Language (CARLa) statements. CARLa is an IBM Security zSecure programming language to create reports for RACF, SMF, UNIX System Services (USS) analysis, and RACF command generation. The sample in Figure 6-1 is to report on only event code 39, which has been caused by a z/OS subsystem (for example, CICS) request failure to map a distributed identity to a RACF user ID.

```

alloc smf
newlist type=smf
select event=RACINIT(39)
sortlist date(9) time(8) type userid,
         event eventqual class intent resource,
         / " "(22) auth_user_name, /* 424 */
         / " "(22) auth_user_regname /* 425 */

```

Figure 6-1 CARLa statements to report direct from active SMF dataset

To capture auditable information concerning identity propagation, SMF has the capability to store this type of information. For SMF Type 80 records, two relocate sections now exist (Table 6-2).

Table 6-2 Additions to RACF SMF Type 80 relocates

Data type (SMF80TP2)	Data length (SMF80DL2)	Format	Audited by event code	Description (SMF80DA2)
424 X'1A8'	1-246	UTF-8	All, except 68,71,79,81, and 82	Authenticated distributed identity user name
425 (x'1A9')	1-256	UTF-8	All, except 68,71,79,81, and 82	Authenticated distributed identity user registry

With the advent of the RACMAP command, event code 87 X'57' occurs to audit the use of this command. The audit occurs with the following usage:

- ▶ UAUDIT
- ▶ SETROPTS SAUDIT
- ▶ SETROPTS AUDIT(USER)
- ▶ SETROPTS CMDVIOL

The two relocate sections listed in Table 6-3 exist and, like the above SMF changes, they are used to hold the long values specified for the user and registry information (Table 6-3).

Table 6-3 Additions to SMF Type 80 relocates for event 87

Data type (SMF80TP2)	Data length (SMF80DL2)	Format	Audited by event code	Description (SMF80DA2)
415 X'19F'	1-246	EBCDIC	87	Value of the user ID filter name from the USERDIDFILTER keyword of RACMAP
416 (x'1A0')	1-256	EBCDIC	87	Value of the registry name from the REGISTRY keyword of RACMAP

For SMF Type 83 we have similar relocate sections existing. SMF Type 83 is another record that records auditing events. These relocate sections will apply to subtype 2 and above. Table 6-4 lists the relocate sections.

Table 6-4 Additions to RACF SMF Type 83 relocates

Data type (SMF83TP2)	Data length (SMF83DL2)	Format	Audited by event code	Description (SMF80DA2)
14 X'E'	1-246	UTF-8	All, except 68,71,79,81, and 82	Authenticated distributed identity user name
15 X'F'	1-256	UTF-8	All, except 68,71,79,81, and 82	Authenticated distributed identity user registry

6.2.2 SMF unload utility

This utility is to extract data from the live SMF datasets and produce an unloaded copy of SMF data for data analysis and reporting. We now have a new record extension mapping. This is to support the RACMAP command (Table 6-5). This shows that after a RACMAP command has been issued, the auditable information from that command can be extracted into a SMF unload file.

Table 6-5 Record extension support for RACMAP command

Format of the RACMAP command extension					
Field name	Type	Length	Start position	End position	Comments
RACM_USER_NAME	Character	20	282	301	Name associated with the user ID.
RACM_UTK_ENCR	Yes/No	4	303	306	Is the UTKEN associated with this user encrypted?
RACM_CTX_MECH	Character	16	2975	2990	Authenticated user authentication mechanism object identifier (OID).
RACM_IDID_USER	Character	985	2992	3976	Authenticated distributed user name.
RACM_IDID_REG	Character	1021	3978	4998	Authenticated distributed user registry name.

Apart from RACMAP, consideration is to be given to the normal operation of RACF and its interaction with distributed identities as requests come in for access. Table 6-6 lists names and positions for SMF data unload to support distributed identity as RACF event code 01.

Note: Events 01 to 89 all have the same record format and information.

Table 6-6 Record extension support for job initiation

Format of the job initiation record extension (event code 01)					
Field name	Type	Length	Start position	End position	Comments
Existing last entry				2530	
INIT_IDID_USER	Character	985	2532	3516	Authenticated distributed user name
INIT_IDID_REG	Character	1021	3518	4538	Authenticated distributed user registry name

6.3 Reporting from SMF data

A variety of software products are able to read this data and provide audit information. We show a few here with examples of their reports.

6.3.1 SMF unload utility

This utility creates records that show audit information for each type of auditable event.

A sequential file is produced by the RACF SMF unload utility (IRRADU00). The contents of this file can be restricted to audit information relating to security events from within RACF. This file can then be processed in a variety of ways:

- ▶ View the file directly.
- ▶ Sort it.
- ▶ Merge it with other data.
- ▶ As an input to other software products that can perform analysis and reporting functions.
- ▶ It can be handled by a Database Manager and stored there. For example, DB2 can provide queries and reports containing this audit information.

The SMF data unload utility has the ability to unload SMF data in the following formats:

- ▶ Extracted data in a sequential file for subsequent processing.
- ▶ Tabular format, which is used to import into a relational database manager.
- ▶ Produces an eXtensible Markup Language (XML) document. This formatted data is then available for rendering onto web pages or processing by other applications sensitive to this XML layout.

Figure 6-2 shows an example of the Job Control Language (JCL) used to perform such an unload. It is designed to show that only SMF Type 80 records are to be unloaded, as they contain RACF auditing information relevant to identity propagation. This example in Figure 6-2 will produce a sequential file suitable for input to other programs to perform analysis and reporting.

```

//AUDIT   JOB   , 'ACQUIRE AUDIT DATA',MSGLEVEL=(1,1)
//SMFDUMP EXEC PGM=IFASMFDP
//SYSPRINT DD  SYSOUT=A
//ADUPRINT DD  SYSOUT=A
//OUTDD   DD   DISP=SHR,DSN=ITSO.AUDIT80.DATA
//SMFDATA DD   DISP=SHR,DSN=SYS1.ITSO.MAN1
//SMFOUT  DD   DUMMY
//SYSIN   DD   *
           INDD(SMFDATA,OPTIONS(DUMP))
           OUTDD(SMFOUT,TYPE(000:255))
           ABEND(NORETRY)
           USER2(IRRADU00)
           USER3(IRRADU86)
/*

```

Figure 6-2 JCL to use the SMF unload utility for SMF type 80 records

6.3.2 SMF UNLOAD produces XML data

This SMF utility can produce an XML file containing unloaded SMF data in an XML format.

It can be produced in a compressed format or a normal layout, depending on which DD is used in the JCL. In our sample (Figure 6-3), we are producing an XML file in a normal format. This file can be analyzed and reported upon by processes provided by the user.

```

//SMFPROP EXEC PGM=IFASMFDP
//SYSPRINT DD  SYSOUT=*
//ADUPRINT DD  SYSOUT=*
//XMLFORM DD DISP=(NEW,CATLG,KEEP),
//          DSN=<name of file to XML data in uncompressed form>
//          SPACE=(CYL,(20,5),RLSE),UNIT=SYSDA,
//          DCB=(LRECL=12288,RECFM=VB,BLKSIZE=0)
//SMFDATA DD DISP=SHR,DSN=<insert name of current active SMF DSN>
//SMFOUT  DD   DUMMY
//SYSIN   DD   *
           INDD(SMFDATA,OPTIONS(DUMP))
           OUTDD(SMFOUT,TYPE(30(1,5),80:83))
           ABEND(NORETRY)
           USER2(IRRADU00)
           USER3(IRRADU86)
/*

```

Figure 6-3 Sample JCL to produce XML from SMF

In Figure 6-3 on page 61, if the XMLFORM DD name is replaced by a XMLOUT DD name, then the XML output will be in compressed form. Figure 6-4 shows an example of uncompressed out.

```

<event>
  <eventType>JOBINIT</eventType>
  <eventQual>DIDNOTDF</eventQual>
  <systemSmfid>SC58</systemSmfid>
  ...
  <details>
    <violation>Y</violation>
    ...
    <evtUserId>STC</evtUserId>
    ...
    <logRacinit>Y</logRacinit>
    ...
    <jobName>SC58CIC1</jobName>
    <readTime>20:30:18</readTime>
    <readDate>2011-01-18</readDate>
    ...
    <ididUser>&#x75;&#x69;&#x64;&#x3D;&#x6D;&#x61;
      &#x72;&#x74;&#x69;&#x6E;&#x61;&#x2C;
      &#x6F;&#x75;&#x3D;&#x73;&#x77;&#x67;
      &#x2C;&#x6F;&#x3D;&#x69;&#x62;&#x6D;
      uid=martina,ou=swg,o=ibm</ididUser>
    <ididReg>&#x77;&#x74;&#x73;&#x63;&#x35;&#x38;
      &#x2E;&#x69;&#x74;&#x73;&#x6F;&#x2E;
      &#x69;&#x62;&#x6D;&#x2E;&#x63;&#x6F;
      &#x6D;&#x3A;&#x33;&#x38;&#x39;
      wtsc58.itso.ibm.com:389</ididReg>
    </details>
  </event>

```

Figure 6-4 Snippets from an uncompressed XMLFORM dataset

The following points relate to the numbers in Figure 6-4:

1. The value DIDNOTDF represents an event qualifier value of 39, indicating that no RACF user ID was found for the distributed ID entity.
2. The violation tag indicates a positive value to reflect an event qualifier of 39.
3. We have logged this activity from a RACINIT call, which reflects that a CICS system has issued a RACROUTE REQUEST=VERIFY to validate a distributed identity.
4. This is a UTF-8 hex representation of the distributed identity.
5. An EBCDIC representation of the distributed identity.

This XML snippet is designed to show what information is recorded when a distributed identity fails to map to a RACF user ID.

Member IRRSCHEM in SYS1.SAMPLIB holds the XML schema document.

6.3.3 Reporting on SMF audit information from DB2

We can extract information from the SMF datasets and load into DB2 and then perform queries on information in the DB2 tables.

First, we put the SMF data into a format suitable for loading into DB2. We use the SMF unload utility to create an output file in which all the fields from the SMF records appear in character format separated by spaces. Figure 6-5 shows a sample JCL deigned to do this.

```
//SMF4DB2 EXEC PGM=IFASMFDP
//SYSPRINT DD SYSOUT=*
//ADUPRINT DD SYSOUT=*
//OUTDD DD DISP=(NEW,CATLG,KEEP),
// DSN=<name of file to hold data in character form>
// SPACE=(CYL,(20,5),RLSE),UNIT=SYSDA,
// DCB=(LRECL=12288,RECFM=VB,BLKSIZE=0)
//SMFDATA DD DISP=SHR,DSN=<insert name of current active SMF DSN>
//SMFOUT DD DUMMY
//SYSIN DD *
        INDD(SMFDATA,OPTIONS(DUMP))
        OUTDD(SMFOUT,TYPE(30(1,5),80:83))
        ABEND(NORETRY)
        USER2(IRRADU00)
        USER3(IRRADU86)
/*
```

Figure 6-5 Sample JCL to extract SMF data and store it in character form

The JCL in Figure 6-5 will use the IRRADU00 exit to write SMF data into the OUTDD dataset in character format (Figure 6-6). As depicted, the column layout lends itself well to loading into DB2 tables.

ADDUSER	SUCCESS	10:28:14	2011-01-20	SC58	NO	NO	NO	ROGERIO	SYS1	NO
RACMAP	SUCCESS	10:28:36	2011-01-20	SC58	NO	NO	NO	ROGERIO	SYS1	NO
SETROPTS	SUCCESS	10:28:41	2011-01-20	SC58	NO	NO	NO	ROGERIO	SYS1	NO
JOBINIT	RACINITD	10:28:48	2011-01-20	SC58	NO	NO	NO	SWGRES	REDBOK02	NO
SETROPTS	SUCCESS	10:29:02	2011-01-20	SC58	NO	NO	NO	ROGERIO	SYS1	NO
ACCESS	SUCCESS	10:29:18	2011-01-20	SC58	NO	NO	NO	STC	TSO	YES

Figure 6-6 Portion of the OUTDD dataset created in Figure 6-5

RACF supplies in SYS1.SAMPLIB a member called IRRADUTB, which defines a table space and tables to contain data as shown in Figure 6-6. Another sample member IRRADULD has the data definition language (DDL) statements to load this unloaded SMF data into DB2 tables.

Having done this, how can we utilize this DB2 data? RACF supplies a member called IRRADUQR in SYS1.SAMPLIB. This member has a number of structured query language (SQL) statements to run against these tables.

In the next example in Figure 6-7, a SQL select occurs against a DB2 table holding RACF JOBINIT records. These records show a failure to map a distributed identity to a RACF user ID, hence the value 'DIDNOTDF'. From the columns we show the EBCDIC values for the distributed identity. The table also holds their UTF-8 equivalents.

```

-----+-----+-----+-----+-----+-----+-----+
SELECT
  *
FROM
  BOBMCC.JOBINIT
WHERE
  INIT_EVENT_QUAL NOT IN ('SUCCESS','TERM','RACINITI','RACINITD')
  ;
-----+-----+-----+-----+-----+-----+-----+
INIT_EVENT_TYPE  INIT_EVENT_QUAL  INIT_IDID_REG_EBCDIC  INIT_IDID_REG_EBCDIC
-----+-----+-----+-----+-----+-----+
JOBINIT          DIDNOTDF         id=martina,ou=swg,o=ibm  wtsc58.itso.ibm.com:389
JOBINIT          DIDNOTDF         ID=MARTINA,OU=SWG,O=IBM  wtsc58.itso.ibm.com:389
JOBINIT          DIDNOTDF         ID=MARTINA,OU=SWG,O=IBM  wtsc58.itso.ibm.com:389

```

Figure 6-7 SQL report showing distributed identities failing to map

Let us consider making an inquiry where we know for a specific application that a user ID of SWGDE is used. Can we see what distributed identities used that application? In the example in Figure 6-8 we see several successful accesses.

```

-----+-----+-----+-----+-----+-----+-----+
SELECT
  *
FROM
  BOBMCC.ACCESS
WHERE
  ACC_EVT_USER_ID = 'SWGDE '
  ;
-----+-----+-----+-----+-----+-----+-----+
ACC_EVENT_TYPE  ACC_EVENT_QUAL  ACC_IDID_USER_EBCDIC  ACC_IDID_REG_EBCDIC
-----+-----+-----+-----+-----+-----+
ACCESS          SUCCESS         UID=MARTINA,OU=SWG,O=IBM  wtsc58.itso.ibm.com:389
ACCESS          SUCCESS         UID=MARTINA,OU=SWG,O=IBM  wtsc58.itso.ibm.com:389
ACCESS          SUCCESS         uid=martina,ou=swg,o=ibm  wtsc58.itso.ibm.com:389

```

Figure 6-8 SQL report showing ACCESS occurrences for user ID SWGDE

6.3.4 Using ICETOOL from DFSORT

ICETOOL is a program that extracts, sorts, and reports from unloaded SMF data. It requires ICETOOL control and reporting statements along with DFSORT statements in separate inputs.

A good ICETOOL reference can be found in the *DFSORT: ICETOOL Mini-User Guide*, which can be found on the DFSORT home page at:

<http://www.ibm.com/storage/dfsort>

Click **publications** → **Papers about DFSORT, ICETOOL and RACF**. On this same page you will also find *RACFIC2 - Security Analysis using RACF Unload Utilities and DFSORT's ICETOOL*.

Let us examine several scenarios and use ICETOOL to provide reports. First, let us consider a system administrator wanting to see whether any of the distributed identities failed to map to a RACF user ID. To do this we must convert the unloaded SMF data file to a file in a more easily digestible form (Table 6-6 on page 60). This file is then supplied as input to the ICETOOL program. Figure 6-9 shows an example of the JCL to run the ICETOOL program.

```
//MAPPINGS EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//PRINT DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//REPORT DD SYSOUT=*
//SMFCHAR DD DISP=SHR,DSN=ITSO.SMF.CHAR.FORMAT -->1
//TEMP001 DD DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,15,0)),
// UNIT=SYSALLDA,VOL=SER=TARTS5, -->2
// DCB=(LRECL=8000,RECFM=VB,BLKSIZE=0)
//TOOLIN DD * -->3
SORT FROM(SMFCHAR) TO(TEMP001) USING(IDPR) -->4
DISPLAY FROM(TEMP001) LIST(PRINT) WIDTH(255) - -->5
PAGE -
TITLE('Failed Mappings') - -->6
HEADER('Date') ON(32,10,CH) -
HEADER('Time') ON(23,8,CH) -
HEADER('User') ON(63,8,CH) -
HEADER('Outcome') ON(14,8,CH) -
HEADER('IDID Name') ON(2783,25,CH) -
HEADER('REG Name') ON(3778,25,CH)
//IDPRCNTL DD *
SORT FIELDS=(23,8,CH,A,32,10,CH,A) -->7
INCLUDE COND=(5,8,CH,EQ,C'JOBINIT ',AND,14,8,CH,NE,C'SUCCESS',AND, -->8
14,7,CH,NE,C'RACINIT')
OPTION VLSHRT
/*
```

Figure 6-9 Sample JCL to produce a report for failed mappings

The comments from the JCL in Figure 6-9 are explained here:

1. The SMFCHAR DD holds the input data in a character format.
2. TEMP001 DD is a temporary dataset to hold the sorted and extracted data from the SMFCHAR DD.
3. The TOOLIN DD holds the ICETOOL statements.
4. This statement tells us that the data in SMFCHAR is to be processed as per the SORT control statements in the IDPRCNTL file. It is implicit that CNTL will be appended to IDPR to form the location where the SORT CONTROL statements are to be found.
5. A statement continued over several lines, governing the layout and columnar data of the report.
6. The title of the report.

7. Sort the extracted data into time and date order.
8. Extract data where it is JOBINIT event and it was not successful and also not a successful RACINIT initiation or deletion.

We will explain how we derived the field position values in the HEADER, SORT, and INCLUDE statements. One approach is to refer to the DDL statements in member IRRADULD within SYS1.SAMPLIB. We have a portion in Figure 6-10 for the JOBINIT records and some field layouts. For HEADER, SORT, or INCLUDE statements, increment the field positions by four to account for the record descriptor word (RDW) at the beginning of the SMFCHAR records.

```

.
.
.
WHEN(1:8)='JOBINIT ' (
  INIT_EVENT_TYPE      POSITION(1:8)      CHAR(8),
  INIT_EVENT_QUAL      POSITION(10:17)    CHAR(8),
  INIT_TIME_WRITTEN    POSITION(19:26)    TIME    EXTERNAL(8)
    NULLIF(INIT_TIME_WRITTEN =' '),
  INIT_DATE_WRITTEN    POSITION(28:37)    DATE    EXTERNAL(10)
    NULLIF(INIT_DATE_WRITTEN =' '),
  INIT_SYSTEM_SMFID    POSITION(39:42)    CHAR(4),
  INIT_VIOLATION       POSITION(44:44)    CHAR(1),
  INIT_USER_NDFND      POSITION(49:49)    CHAR(1),
  INIT_USER_WARNING    POSITION(54:54)    CHAR(1),
  INIT_EVT_USER_ID     POSITION(59:66)    CHAR(8),
.
.
.

```

Figure 6-10 Portion of DDL statements relating to identity propagation

The purpose of the job as detailed in Figure 6-9 on page 65 is to provide a report for all RACF user IDs that start with SWG. For the purposes of this report let us assume that any RACF user ID that begins with SWG is to be used by distributed identities. Figure 6-11 shows a portion of this report.

Note that this report has been compressed to remove spaces between columns of data normally placed there by ICETOOL.

1- Failed Mappings					
Date	Time	User	Outcome	IDID Name	REG Name
2011-01-27	10:06:16	STC	DIDNOTF	uid=martina,ou=swg,o=ibm	wtsc58.itso.ibm.com:389
2011-01-25	16:29:17	STC	DIDNOTF	UID=MARTINA,OU=SWG,O=IBM	wtsc58.itso.ibm.com:389
2011-01-19	16:42:52	STC	DIDNOTF	uid=martina,ou=swg,o=ibm	wtsc58.itso.ibm.com:389
2011-01-19	16:43:27	STC	DIDNOTF	uid=bob,ou=swg,o=ibm	wtsc58.itso.ibm.com:389

Figure 6-11 Distributed identities that failed to map to a RACF user ID

Another scenario is to see what distributed identities gained access to a z/OS subsystem and who were they. In our example, we look for any successful access for RACF user IDs beginning with SWG. Figure 6-12 depicts the JCL that we used.

```
//ACCESSES EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//PRINT DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//REPORT DD SYSOUT=*
//SMFCHAR DD DISP=SHR,DSN=ITSO.SMF.CHAR.FORMAT
//TEMPO001 DD DISP=(NEW,DELETE,DELETE),SPACE=(CYL,(30,15,0)),
// UNIT=SYSALLDA,VOL=SER=TARTS5,
// DCB=(LRECL=8000,RECFM=VB,BLKSIZE=0)
//TOOLIN DD *
SORT FROM(SMFCHAR) TO(TEMPO001) USING(IDPR)
DISPLAY FROM(TEMPO001) LIST(PRINT) WIDTH(255) -
PAGE -
TITLE('Distributed Identity Accesses ') -
HEADER('Date') ON(32,10,CH) -
HEADER('Time') ON(23,8,CH) -
HEADER('User') ON(63,8,CH) -
HEADER('Outcome') ON(14,8,CH) -
HEADER('IDID Name') ON(3597,25,CH) -
HEADER('REG Name') ON(4592,25,CH)

/*
//IDPRCNTL DD *
SORT FIELDS=(23,8,CH,A,32,10,CH,A) -->1
INCLUDE COND=(5,8,CH,EQ,C'ACCESS ',AND,63,3,CH,EQ,C'SWG') -->2
OPTION VLSHRT
/*
```

Figure 6-12 Sample JCL to produce report of successful accesses

Figure 6-12 is explained further here:

1. The selected records are sorted in time and date order.
2. Records are selected if the outcome field has the value ACCESS and the user responsible begins with SWG.

In Figure 6-13 we now have a list of distributed entities along with their RACF user IDs. Note that this report has had blank columns removed in order to fit in the figure. We chose only a small amount of data to show the nexus between distributed identity and the RACF user ID. In this example the ratio is 1:1.

- 1 - Distributed Identity Accesses						
Date	Time	User	Outcome	IDID Name	REG Name	
2011-01-20	10:09:40	SWGRES	SUCCESS	UID=MARTINA,OU=SWG,0=IBM	wtsc58.itso.ibm.com:389	
2011-01-20	10:23:48	SWGRES	SUCCESS	UID=MARTINA,OU=SWG,0=IBM	wtsc58.itso.ibm.com:389	
2011-01-24	11:25:05	SWGDE	SUCCESS	uid=martina,ou=swg,o=ibm	wtsc58.itso.ibm.com:389	
2011-01-19	16:49:35	SWGGAU	SUCCESS	UID=BOB,OU=SWG,0=IBM	wtsc58.itso.ibm.com:389	
2011-01-19	16:52:54	SWGGAU	SUCCESS	UID=BOB,OU=SWG,0=IBM	wtsc58.itso.ibm.com:389	

Figure 6-13 List of distributed identities along with their RACF user IDs

In Figure 6-13 on page 67, the two IDID names UID=MARTINA,OU=SWG,O=IBM and uid=martina,ou=swg,o=ibm are not identical. You will see that they map to different RACF user IDs.

6.3.5 Using IBM Security zSecure Audit for RACF

IBM Security zSecure Admin provides security personnel with tools to help unleash the potential of your mainframe system, enabling efficient and effective RACF administration, while helping use fewer resources.

By automating many recurring system administration functions and by enhancing the native RACF authorization and delegation capabilities, Security zSecure Admin can help you maximize IT resources, reduce errors, improve quality of services, and demonstrate compliance. It provides its own process to extract data from SMF files. A sample is supplied within the sample library SCKRSAMP. The sample JCL (C2RJFUNL) can extract a range of SMF records. In Figure 6-14 we observe only SMF Type 80 being extracted. The statements in this SYSIN are CARLa statements that are used with the zSecure product to programmatically carry out requested actions.

```
//SYSIN DD *
  alloc smf
  suppress CKFREEZE
  n type=smf name=smfse1
  s type=(80)
  unload dd=smfunld
  include member=CKALFSUM
```

Figure 6-14 SYSIN statements in CARLa programming language

When this amended sample JCL is run, the SYSPRINT file will show what SMF files were read and how many records were selected (Figure 6-15).

```
CKR0450 00 Started processing SMF file CKR@SM00 TARCAT SYS1.SC58.MAN1
CKR0450 00 Started processing SMF file CKR@SM01 TARCAT SYS1.SC58.MAN2
CKR0450 00 Started processing SMF file CKR@SM02 TARCAT SYS1.SC58.MAN3
CKR0427 00 1,064 SMF records read, 99 SMF records selected (9%)
```

Figure 6-15 SYSPRINT showing files read and number of records extracted

At this point we have a data file ready for data analysis and reporting. We have a range of methods to do this.

Examining SMF data using ISPF panels with zSecure

By using the zSecure panel we can:

- ▶ See who, when, and where a distributed identity was created.
- ▶ Examine audit information showing access violations for a distributed identity.
- ▶ Look at access warnings issued for a distributed identity.

After entering the zSecure main menu, you will need to check that there is a current CKFREEZE file. Then using option SE.1 go into this zSecure panel and add the new SMF unloaded files to the list of selected files. This process is not explained here, as zSecure users will be familiar with it.

Using some of these displays, let us see what information we can obtain and how it relates to auditing of distributed identity. We base our discussion on the EVENTS option on the zSecure main menu.

1. Select **EV.1** for SMF Reports 'predefined analysis reports'.
2. Select **Exceptions** for 'RACF exception report'.
3. Enter a value if you want to reduce the volume of SMF data analyzed or press Enter.

CMDSPEC commands issued by SPECIAL users

This section shows RACF commands issued by RACF user IDs with the SPECIAL attribute. In our example in Figure 6-16 we show user ITS01 issuing RACMAP commands and then doing a refresh of the IDIDMAP class. This display shows a number of IDIDMAP profiles being created and deleted.

```

Commands issued by SPECIAL users                                     Line 1 of 21
Command ===>                                                       Scroll===> PAGE
                                                                    19Jan11 16:31 to 21Jan11 09:21

  User      Full Name          Count
  ITS01     ITS01 SPECUSR         12
  Date/time      RACF command
  19Jan11 17:01:55.35 RACMAP ID(NORACMAP) MAP WITHLABEL('LABEL00000001')
  19Jan11 17:02:02.82 SETROPTS RACLIST(IDIDMAP) REFRESH
  19Jan11 17:16:08.18 RACMAP ID(SWGRES) MAP WITHLABEL('LABEL00000001')
  19Jan11 17:16:17.69 SETROPTS RACLIST(IDIDMAP) REFRESH
  19Jan11 17:24:50.19 RACMAP ID(SWGRES) MAP WITHLABEL('LABEL00000002')
  19Jan11 17:24:57.52 SETROPTS RACLIST(IDIDMAP) REFRESH
  19Jan11 17:27:20.97 RACMAP ID(SWGRES) DELMAP LABEL('LABEL00000002')
  19Jan11 17:27:24.53 SETROPTS RACLIST(IDIDMAP) REFRESH
  19Jan11 17:27:37.73 RACMAP ID(SWGRES) DELMAP LABEL('LABEL00000001')
  19Jan11 17:27:40.50 SETROPTS RACLIST(IDIDMAP) REFRESH
  19Jan11 17:27:50.73 RACMAP ID(SWGRES) MAP WITHLABEL('LABEL00000001')
  19Jan11 17:27:55.05 SETROPTS RACLIST(IDIDMAP) REFRESH
***** Bottom of Data *****

```

Figure 6-16 A zSecure display of commands issued by a SPECIAL user ID

How can this be of use to us? First, we have a list of commands relevant to setting up mappings for a distributed user, so this is an audit trail of who created these mappings (that is, creating the nexus between a RACF user ID and one or more distributed identities).

We can issue a select against any of these commands to obtain more information about a particular command and the distributed identity used. Select the RACMAP command for SWGRES and a snippet of the subsequent screen is shown. Figure 6-17 shows a RACF ID of SWGRES mapped to a distributed identity. Also, it tells us that the command was successful.

```

Commands issued by SPECIAL users                               Line 1 of 41
Command ===>                                                Scroll===> PAGE
                                                                19Jan11 16:31 to 21Jan11 09:21

RACF command
RACMAP ID(SWGRES) MAP WITHLABEL('LABEL00000002')
USERDIDFILTER(NAME('UID=MARTina,OU=SWG,O=IBM'))
REGISTRY(NAME('*'))

Record identification
Job name + id          ITS01
SMF date/time         Wednesday 19 Jan 2011 17:24:50.19
System ID             SC58          record no: CKR1SM01 1180

Event identification
RACF event description Distributed ID map (Success:Successful RACMAP)
RACF event qualifier   0
RACF descriptor for event Success
RACF reason for logging Special
SAF authority used     Special

Subject identification
User: ITS01          Group: SYS1          Terminal: SC38TC6B  Appl:
Name: SPECUSR              Security label:
Token: User:ITS01; Group:SYS1; Flags:(Pre 1.9,Special); Session:TS0;
Port:TERMINAL(SC38TC6B)

```

Figure 6-17 A display of SMF data relating to a RACMAP command

GRESVIOL general resource access violations by class, profile, and user

In the example in Figure 6-18 it is observed that an access violation has occurred. In this case a distributed identity is being checked by CICS after its credentials have been supplied by the CICS Transaction Gateway (CICS TG) program. At this point there is no difference whether the user is a normal user or has a distributed identity.

```

General Resource Access Violations by Class, Profile and User   Line 1 of 7
Command ===>                                                Scroll===> PAGE
                                                                19Jan11 16:31 to 21Jan11 09:21

Class   Prof#  Users  Count
TCICSTRN   1    1    1
Profile
CSMI                               Users  Count
                               1    1
User   Full Name          Count
RACMAP01 DEFAULT RACMAP          1
Date/time          Description
19Jan11 17:11:33.22 RACF ACCESS violation for RACMAP01: (READ,NONE) on TCICSTRN
CSMI

```

Figure 6-18 CICS mirror transaction experiences an access violation

By issuing a select command against this ACCESS violation we have a more detailed view of this occurrence. Figure 6-19 is only a portion of the display designed to show only the most important information.

```

Description
RACF ACCESS violation for RACMAP01: (READ,NONE) on TCICSTRN CSMI

Record identification
Job name + id           SC58CIC1
SMF date/time          Wednesday 19 Jan 2011 17:11:33.22
System ID              SC58           record no: CKR1SM01 1140

Event identification
RACF event description  Resource access (Failure:Insufficient
RACF event description authority)
RACF event qualifier    1
RACF descriptor for event Violation
RACF reason for logging Resource
SAF authority used      Normal
Access intent           READ
Access allowed          NONE
Audit/message logstring

Object identification
SAF profile class       TCICSTRN
SAF profile key         CSMI
SAF resource name       CSMI
Volume serial

Subject identification
User: RACMAP01      Group: REDB0K02      Terminal:           Appl: SC58CIC1
Name: DEFAULT RACMAP      Security label:
Token: User:RACMAP01; Group:REDB0K02

Authenticated user identity mapping
Authenticated user name  UID=MARTINA,OU=SWG,O=IBM
Authenticated user regname wtsc58.itso.ibm.com:389

```

Figure 6-19 Detailed information about an ACCESS violation

We can tell that RACF user ID RACMAP01 has a distributed identity. With this knowledge, the security administrator can make the determination of what action to follow. In this case, it would be to grant access to this CICS transaction.

GRESWARN general resource access warnings by class, profile, and user

If we continue on from the previous example, the security administrator grants access to the CSMI transaction for our distributed identity but sets a warning for this access. So when our distributed identity enters CICS, RACF will write data to SMF about this access to CSMI. This SMF data is displayed in the portion of the screen shown in Figure 6-20.

Description			
RACF ACCESS warning for RACMAP01: (READ,NONE) on TCICSTRN CSMI			
Record identification			
Job name + id	SC58CIC1		
SMF date/time	Wednesday 19 Jan 2011 17:09:11.54		
System ID	SC58	record no:	CKRISM01 1128
Event identification			
RACF event description	Resource access (Warning:Access permitted due to warning)		
RACF event description	to warning)		
RACF event qualifier	3		
RACF descriptor for event	Warning		
RACF reason for logging	Resource		
SAF authority used	Normal		
Access intent	READ		
Access allowed	NONE		
Audit/message logstring			
Object identification			
SAF profile class	TCICSTRN		
SAF profile key	CSMI		
SAF resource name	CSMI		
Volume serial			
Subject identification			
User: RACMAP01	Group: REDBOK02	Terminal:	Appl: SC58CIC1
Name: DEFAULT RACMAP		Security label:	
Token: User:RACMAP01; Group:REDBOK02			
Authenticated user identity mapping			
Authenticated user name	UID=MARTINA,OU=SWG,O=IBM		
Authenticated user regname	wtsc58.itso.ibm.com:389		

Figure 6-20 A distributed identity shows an access warning to a CICS transaction

RACF events: RACF logging for specific events

Another probable scenario involves locating successful initiations by distributed identities. This provides an auditor and the security administrator with information about remote users who perform work on their system.

To examine such data we take the following steps from the zSecure main menu:

1. Select **EV.2 RACF Events**'.
2. Select **'All events'**.
3. Enter * user id.
4. Press Enter to bypass SMF criteria if not required.
5. Press Enter to bypass maximum SMF records criteria if required.
6. Look for successful racinit initiation messages.

Having followed these instructions, we are presented with a screen similar to Figure 6-21. This shows a distributed identity preparing to enter CICS and then about 3 minutes later being validated prior to entering CICS again.

```

SMF record RACF processing and audit records                               Line 1 of 17
Command ===>                                                            Scroll===> PAGE
                                                                           17Jan11 05:30 to 21Jan11 09:21

  Event      Q      Count Event description
  RACINIT  12          17 Racinit (Success:Successful racinit initiation)
  Date/time      Description
  19Jan11 16:49:35.72 RACF RACINIT success for SWGAU: Job Start / Logon,
  APPL=SC58CIC1
  19Jan11 16:52:54.98 RACF RACINIT success for SWGAU: Job Start / Logon,
  APPL=SC58CIC1

```

Figure 6-21 Distributed identities prior to entering CICS

Figure 6-22 is an example of a successful mapping of a distributed identity to a RACF user ID.

```

Description
  RACF RACINIT success for SWGAU: Job Start / Logon, APPL=SC58CIC1

Record identification
  Jobname + id: SC58CIC1
  SMF date/time: Wed 19 Jan 2011 16:49:35.72
  SMF system: SC58          record type: 80  record no: CKR1SM01 804

Event identification
  RACF event description      Racinit (Success:Successful racinit initiation)
  RACF event qualifier        12
  RACF descriptor for event   Success

SAF profile class            USER
SAF profile key              SWGAU
SAF resource name            SWGAU

Subject identification
  User: SWGAU          Group: REDBOK02      Terminal:           Appl: SC58CIC1
  Name: SWGAU          Security label:
  Token: User:SWGAU; Group:REDBOK02

Authenticated user identity mapping
  Authenticated user name      UID=BOB,OU=SWG,O=IBM
  Authenticated user regname   wtsc58.itso.ibm.com:389

```

Figure 6-22 A distributed identity successfully authenticated prior to entering CICS

Using batch or online CARLa statements

Our previous discussion centred on screen-based enquiries where several screens were required to drill down to a specific SMF record with data relevant to a distributed identity.

By way of background CARLa programs can be deployed in the following ways:

- ▶ From batch JCL with CARLa statements being supplied as SYSIN or called via a IMBED statement.
- ▶ Selecting the COMMANDS “Run commands from library” option from the main zSecure menu, where CARLa program can be selected from a library to be run.
- ▶ By entering CARLa at the command prompt on most zSecure panels, a CARLa statement can be entered and run in an iterative manner.

From an administrative and audit viewpoint it is useful to establish a library of CARLa programs to be used for specific purposes.

In this discussion about using CARLa to display auditing information, we start from a summary of audited events and then produce a detailed analysis upon the audit data. Our summary would be produced by the CARLa statements in Figure 6-23. This program needs a SMF001 DD pointing at one or more unloaded SMF datasets.

```
suppress msg=1400
suppress racf ioconfig
alloc type=smf DD=SMF001
newlist type=smf title='1 All Events Comands'
select type=80
summary event count
```

Figure 6-23 CARLa statements to produce a summary of events

Looking at our event summary (that is, what events RACF wrote to SMF), we see a simple scenario with most users being validated and then performing actions against a variety of protected resources that record their access (Figure 6-24).

```
S M F   R E C O R D   L I S T I N G   17Jan11 05:30 to 21Jan11 09:21
1 All Events Comands

Event      Count
RACINIT    23
ACCESS     804
DEFINE     2
CONNECT    1
PERMIT     1
RALTER     3
RDEFINE    1
SETROPTS   9
RACMAP     6
```

Figure 6-24 A summary of RACF events

Let us now produce a report showing validation of a distributed identity and accesses to a CICS system using a CARLa script (Figure 6-25). This contains exclude CARLa statements to remove data not relevant to our purpose.

```

suppress msg=1400
suppress racf ioconfig
alloc type=smf dd=smf001
newlist type=smf TITLE='Distributed Identities in CICS'
select jobname=sc58cic*
exclude userid=stc
exclude resource=(cemt,ceda,ec01,ec02,ec03,ec04,cedf,cedx)
sortlist datetime(19) jobname userid,
        event eventqual class intent resource(8),
        / auth_user_name(23),
        auth_user_regname(23)

```

Figure 6-25 CARLa script to extract and report on distributed identities

The report output shown in Figure 6-26 shows:

1. A distributed identity entering z/OS.
2. Being validated and mapped (RACINIT 12) to a RACF user ID MARTINA.
3. Being given access into CICS.
4. Being timed out at 12:27 because CICS user delay is 5 minutes.

S M F R E C O R D L I S T I N G 19Jan11 11:43 to 28Jan11 12:42							
Distributed Identities in CICS							
Date/time	Jobname	User	Event	Eq	Class	Intent	Resource
28Jan11 12:22:53.35	SC58CIC1	MARTINA	RACINIT	12	USER		MARTINA
UID=MARTINA,OU=SWG,0=IB wtsc58.itso.ibm.com:389							
28Jan11 12:22:53.35	SC58CIC1	MARTINA	ACCESS	1	TCICSTRN	READ	CSMI
UID=MARTINA,OU=SWG,0=IB wtsc58.itso.ibm.com:389							
28Jan11 12:27:53.37	SC58CIC1	MARTINA	RACINIT	13	USER		MARTINA
UID=MARTINA,OU=SWG,0=IB wtsc58.itso.ibm.com:389							


Figure 6-26 Report on distributed identity entering a CICS system called SC58CICS

In our next report (Figure 6-27) we see a distributed identity moving from one CICS to another CICS system (for example, from a terminal owning region (TOR) to an application owning region (AOR)):

1. A distributed identity entering z/OS.
2. Being validated and mapped (RACINIT 12) to a RACF user ID SWGDE.
3. Being given access into CICS (TOR).
4. Being given access into the second CICS (AOR).
5. Being timed out at 12:42 because the first CICS user delay is 5 minutes.
6. Being timed out at 12:42 because the second CICS user delay is 5 minutes.

S M F R E C O R D L I S T I N G 28Jan11 12:30 to 28Jan11 12:42							
Distributed Identities in CICS							
Date/time	Jobname	User	Event	Eq	Class	Intent	Resource
28Jan11 12:37:46.16	SC58CIC1	SWGDE	RACINIT	12	USER		SWGDE
uid=martina,ou=swg,o=ib wtsc58.itso.ibm.com:389							
28Jan11 12:37:46.16	SC58CIC1	SWGDE	ACCESS	0	TCICSTRN	READ	CSMI
uid=martina,ou=swg,o=ib wtsc58.itso.ibm.com:389							
28Jan11 12:37:46.34	SC58CIC2	SWGDE	ACCESS	0	TCICSTRN	READ	CSMI
uid=martina,ou=swg,o=ib wtsc58.itso.ibm.com:389							
28Jan11 12:42:46.50	SC58CIC1	SWGDE	RACINIT	13	USER		SWGDE
uid=martina,ou=swg,o=ib wtsc58.itso.ibm.com:389							
28Jan11 12:42:47.02	SC58CIC2	SWGDE	RACINIT	13	USER		SWGDE
uid=martina,ou=swg,o=ib wtsc58.itso.ibm.com:389							

Figure 6-27 Report on a distributed identity traversing several CICS systems



Internal z/OS data structures impacted by identity propagation

This chapter shows what SAF interfaces and data structures have support for identity propagation.

7.1 SAF interfaces

In this section we discuss relevant data areas impacted by identity propagation.

7.1.1 RACF communication vector table

The RACF communication vector table is a communication area for information global to RACF functions. This is fully defined in *z/OS Security Server RACF Data Areas, GA22-7680*. To cater for identity propagation, two fields exist within this programming interface. They hold information that defines the current acceptable maximum length of:

- ▶ User's distinguished name
- ▶ Registry name

This is outlined in the snippet from the data area structure ICHPRCVT shown in Table 7-1.

Table 7-1 Additional fields added to CVT

Offset	Data type	Data length	Field name	Field description
372 X'174'	Unsigned Integer	2	RCVTDNL	Maximum length of distributed user ID - 246 UTF-8 characters.
374 X'176'	Unsigned Integer	2	RCVTRL	Maximum length of registry name - 255 UTF-8 characters.
376 X'178'	Unsigned	1	RCTIDPV	Indicates enhancements provided by APARs OA34258 and OA34259 are available on the system.

7.1.2 RACROUTE

This SAF callable service is invoked in the following manner:

```
RACROUTE REQUEST=VERIFY/VERIFX
```

The following additional parameters are placed on this call:

- ▶ ICRX = *icrx address*
- ▶ IDID = *idid address*

It is the caller's responsibility to create a data area for ICRX and Distributed Identity Data (IDID) unless it has been passed by a calling application. In the later case, it could well be that the Identity Context Reference (ICR) field might contain a value.

One additional parameter that is required is ENVIR=CREATE. This call, if successful, is to create an ACEE. Information about this macro can be obtained from the *Security Server RACROUTE Macro Reference*, SA22-7692. Along with these changes, we provide additional reason codes in Table 7-2.

Table 7-2 Reason codes from RACROUTE when using ICRX

Abend 283 (RACROUTE REQUEST=VERIFY parameter list error)		
Reason code	Short description	Full description
6C	ICRX block is not valid.	Either the ID value or length values were not valid, or the ICRX parameter was specified with the IDID, ICTX, NESTED=COPY, or NESTED=YES parameter.
70	IDID block is not valid.	Either the ID value, subpool, or length values were not valid, or the IDID parameter was specified with the ICTX, NESTED=COPY, or NESTED=YES parameter.

7.1.3 InitACEE

An application calls an initACEE callable service to create an ACEE. As previously mentioned, the ACEE is a control block that defines the runtime security environment. When we make a call to this SAF interface we now also supply a pointer to a IDID data structure. IDID_Area is the name of a fullword containing the address to this data structure.

Some operational hints are:

- ▶ If the Function_Code requests an ACEE to be built and no RACF_UserID parameter is supplied, plus the IDIDMAP class is active and RACLISTed, then information in the IDID is used to determine a RACF user ID.
- ▶ If the IDIDMAP class is inactive or the information in the IDID does not map to a RACF user ID, the InitACEE will fail.
- ▶ If a RACF_UserID parameter is specified on the InitACEE, the IDID_Area parameter can supply the name of an IDID to be associated with the ACEE. The IDIDMAP class is not used.
- ▶ When the IDID_Area parameter is specified, the distributed identity information in the IDID should have been previously authenticated.
- ▶ If an IDID is supplied and an ACEE is successfully created, the ACEE will point to a copy of the IDID, and it will subsequently be used in auditing.

Figure 7-1 shows a sample layout of such a call with the IDID_Area holding the address of the IDID data structure as the last parameter.

```
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 7-1 Sample CALL layout for InitACEE

This service is documented in *z/OS Security Server RACF Callable Services, SA22-7691*.

7.1.4 R_cacheserv

To support identity propagation we use a number of functions through R_cacheserv to store entities into the RACF local identity context cache, retrieve them, and delete them. These entities are called ENVR objects, which might or might not contain an IDID.

This cache is local to the Sysplex, and by its nature it is read/write. Here is example of calling this service:

```
CALL IRRSCH00 (Work_Area, ..., ICRX_Area, ICRX_Length)
```

Management of extended read/write cache is handled by function code 7. With the advent of enhanced functions with OA34258 and OA34259, function code 7 now has five option parameters (Table 7-3). The life for of a cached entity is 60 minutes.

Table 7-3 Options available to function code 7 in R_cacheserv

Option	Purpose	Description
1	Store data and return ICRX.	Returns an extended reference using a new ICRX parameter. It will use the ACEE as the source record and create a record name from the IDID pointed to by the ACEE. A cache reference will be created for the new record, and its ICR will be returned in an ICRX along with the IDID and RACF user ID.
2	Retrieve application data.	Behaves in a similar way to function code 6 option 4 (retrieve application data).
3	Remove record.	Behaves in a similar way to function code 6 option 5 (remove record) if an ICR is set in the ICRX. It will mark a data record as no longer valid if an IDID is provided in the ICRX with a non-set ICR.
4	Store and return reusable ICRX.	Similar processing to option 1, but the ICRX will be marked as reusable. It might called by option 2 multiple times and will time out after an hour of inactivity.
5	Validate input ICRX.	This is a validation function to validate a user-built ICRX that has been provided to a subsystem, for example, CICS. Note that it is not to done on a completed ICRX as returned by RACF as with option 1.

This service is documented in *z/OS Security Server RACF Callable Services*, SA22-7691.

7.1.5 R_usermap

With respect to identity propagation, this function uses a user's distinguished name and a registry/realm name to determine a mapped RACF user ID. This association should already exist following RACMAP commands. Table 7-4 describes function code X'0008'.

Table 7-4 R_usermap

Function code	Purpose	Description
X'0008'	Return RACF user ID that has been associated with the user's distinguished name and registry/realm name.	Provide a query service invoked via a callable service to take the user's distinguished name and registry/realm name and return the matching RACF user ID.

Note: For function x'0008' to use this service, the subsystem can be authorized by the resource IRR.IDIDMAP.QUERY in the FACILITY Class if not running in system key or supervisor state.

This is a new function of R_usermap provided by OA34258 and OA34259.

7.2 SAF data areas

The information mentioned here is covered in more depth in *Security Server RACF Data Areas*, GA22-7680.

7.2.1 Accessor Environment Element

This is an z/OS control block that exists at a z/OS address space level and optionally at the z/OS task level. The Accessor Environment Element (ACEE) is the *run-time security context* that anchors the identity (in RACF terms; that is, the RACF user ID) for whom the address space or task is performing work. It represents the authorities of a single accessor in the address space.

An ACEE is built by SAF-RACF when a server-hosting environment such as CICS asks SAF-RACF to do so via invocation of the RACROUTE Request=Verify SAF callable service.

Two fields have been added at the end of this structure in the ACEE to support the identity propagation:

- ▶ ACEEIDID points to the IDID data area holding the distributed identity data.
- ▶ ACEETIME captures the creation time.

Table 7-5 new ACEE fields

Offset	Data type	Data length	Field name	Field description
184 X'B8'	Address	4	ACEEIDID	Address of distributed data (IDID)
188 X'BC'	Character	4	ACEETIME	ACEE creation time

Look in *z/OS Security Server RACF Data Areas*, GA22-7680, for the mapping of the data structure IHAACEE. A point worth noting here is that the 4-character *ACEE* is changed to *acee* prior to freeing the ACEE storage. Figure 7-2 shows how the ACEE is connected to the IDID.

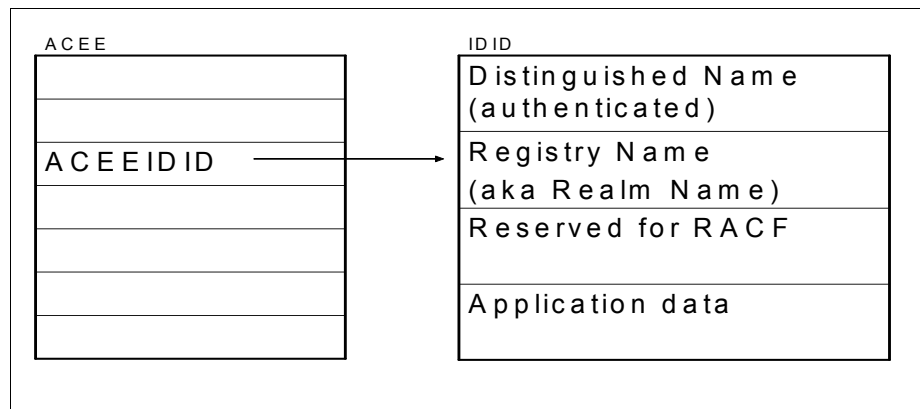


Figure 7-2 ACEE structure and pointer to IDID

7.2.2 ICRX: Extended Identity Context Reference

This holds information needed to retrieve or rebuild an authenticated distributed user's z/OS security environment. It can be found as:

- ▶ Output from R_cacheserv
- ▶ Input into R_cacheserv
- ▶ Input for RACROUTE REQUEST=VERIFY

When passed as an input parameter, it is in the form ICRX=icrx addr. This specifies an address that points to a caller-provided ICRX area.

If creating an ICRX, the caller is responsible for:

- ▶ Setting the ICRX ID, version, subpool, and length.
- ▶ All applicable field length and offset values, unless the ICRX was previously obtained from R_cacheserv. RACF checks the ID and validates that the specified length values in the RACF sections do not exceed the allowed maximum values.
- ▶ Freeing the ICRX structure.

It is mapped by the IRRPICRX data area structure and contains a 4-byte eyecatcher 'ICRX' and has a variable size. It can be created by RACF or a caller to RACF. This ICRX now has a

flag to indicate that it is a multi-use ICRX. This enhancement is supplied by APARs OA34258 and OA34259.

7.2.3 IDID: Distributed Identity Data

It is mapped by the IRRPIDID data area structure and contains a 4-byte eyecatcher 'IDID' and has a variable size. It can be created by RACF or a caller to RACF and its address is held in the ACEEIDID field in the ACEE.

It also holds information pointing to a number of optional sections. They are:

- ▶ z/OS section
- ▶ Reserved for additional RACF security information
- ▶ Reserved for alternative security information
- ▶ Reserved for customer use
- ▶ Reserved for use by Websphere Application Server

From the view of identity propagation, the most important section is the z/OS section. In its structure it has two important fields:

- ▶ User's distinguished name

Distributed client end-user's identity, within registry designated by IDID1REG, in UTF8 form, represented as one of the following:

- LDAP string form of the user's X.500 distinguished name as defined within the LDAP registry, in canonical form as in RFC2253, and with LDAP special characters escaped with a '\' (UTF-8'92'x). Note that this is identical to what is the result of a `WASwscredential.getUniqueSecurityName()` method invocation run on a WebSphere Application Server having authenticated the user by way of the LDAP registry.
- A simple character string such as a user ID as defined within a registry.

- ▶ Registry's name

The name of the original registry in UTF8 format. Note that this is identical to what would be the result of a WebSphere Application Server `wscredential.getRealmName()` method invocation run on a WebSphere Application Server having authenticated the user by way of the LDAP registry.

7.2.4 ENF2: RACF ENF Event Code 71

This maps the input parameter list for ENF event code 71 listen exits. It is mapped by the IRRPENF2 data area structure and contains a 6-byte eyecatcher 'IRREN2' and has a fixed size of 24 bytes. It is created by RACF.



Identity propagation with CICS and CICS Transaction Gateway

This chapter describes a scenario for identity propagation in a remote topology.

8.1 Architectural overview

Figure 8-1 shows the topology that was used for this scenario. CICS Transaction Server Version 4.1 and CICS Transaction Gateway Version 8 are both on z/OS. The distributed identity is held in IBM Tivoli® Directory Server and mapped to a user ID in RACF when it is passed to CICS Transaction Server. This scenario uses WebSphere Application Server Version 7 and the CICS Transaction Gateway (CICS TG) resource adapter on Windows. The distributed authentication was performed using an LDAP server.

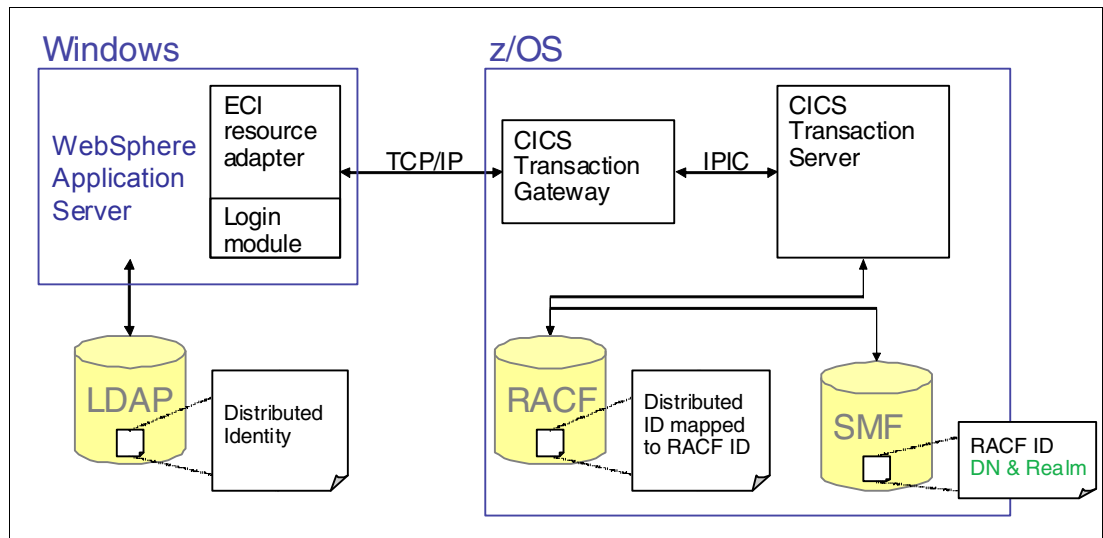


Figure 8-1 CICS identity propagation topology used in this scenario

Figure 8-2 shows a diagram of CICS identity propagation using multiple CICS regions.

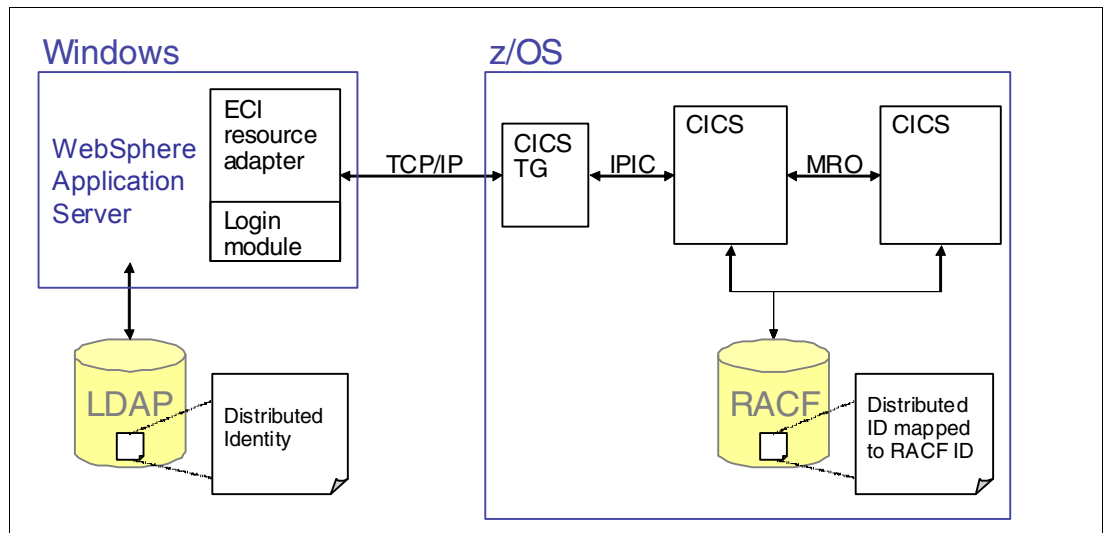


Figure 8-2 CICS identity propagation using multiple CICS regions

To test that the scenario works successfully, the sample application ECIDateTime supplied by CICS Transaction Gateway was used:

- ▶ The ECIDateTime sample EJB application must be installed on WebSphere Application Server.
- ▶ The sample CICS Transaction Gateway server program EC01 must be compiled, defined, and installed in a CICS program library.
- ▶ We also added two programs modelled after EC01 (EC03 and EC04) to enable us to demonstrate the retrieval of DN and realm information within a CICS program, both in the region that CICS TG connects to and in a downstream CICS-to-CICS connected region. These programs write diagnostic information to CICS TD queue CSSL.

8.2 Configuring identity propagation on CICS Transaction Gateway

The connectivity between CICS TG and CICS needs to be via an IPIC connection.

The IPIC protocol was introduced in CICS TS 3.2 and is a TCP/IP-based protocol that offers advantages over LU6.1/APPC SNA protocol. Identity propagation can occur over IPIC connections, but not over LU6.2 connections. For a quick introduction to IPIC, see:

<http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.cicstg/cicstg/7.2z/0verview/IPICIEA/player.html>

The steps in this section describe how to configure CICS TG to pass a distributed identity to CICS. These changes can be made via the TSO ISHELL command to the configuration file used by CICS TG:

```
/ctg/ctg800/ctg.idprop.ini
```

First, you need the following pieces of information:

- ▶ Eight-character string to be used as an APPL identifier. Note this is not a VTAM® applid.
- ▶ Eight-character string to be used as a APPL qualifier. Note this is not a VTAM network ID.
- ▶ Name to use within CICS TG for the CICS connection.
- ▶ DNS name or IP address for the CICS region.
- ▶ Port number that the CICS region will be listening.

To configure CICS TG to pass a distributed identity to CICS:

1. Add a PRODUCT section, which points to the IPIC SERVER section via the default server (Figure 8-3).

```
SECTION PRODUCT
#
applid=MYAPPL
applidqualifier=MYNETID
defaultserver=ITSOLAB
ENDSECTION
```

Figure 8-3 PRODUCT section in CICS TG in file

2. Add an IPIC SERVER section, which communicates with our CICS region that is listening on port 50889 (Figure 8-4).

```
SECTION IPICSERVER=ITSOLAB
#
hostname=wtsc58.itso.ibm.com
port=50889
ENDSECTION
```

Figure 8-4 IPIC server section

Note: You might prefer not to use a default server. In our scenario, we used a default CICS server.

3. In addition to the above configuration steps, we also installed CICS SupportPac CH51 to assist in troubleshooting. This allowed us to verify that a distributed identity was being received by CICS TG from WebSphere Application Server. This would be an optional step at your installation. The SupportPac includes detailed instructions on how to install. The CICS SupportPac Ch51 is available at:

<http://www-01.ibm.com/support/docview.wss?uid=swg24019170>

8.3 Configuring identity propagation on CICS Transaction Server

The following steps describe how to configure CICS to accept a distributed identity. See 8.2, “Configuring identity propagation on CICS Transaction Gateway” on page 87, for information needed before defining these resources.

1. Configure a TCPIP SERVICE resource to listen on the designated port (Figure 8-5):
 - The PORTNUMBER must match the port number defined in CICS TG for the IPIC server.
 - The PROTOCOL must be IPIC for this to work correctly.

```

CEDA View TCpipservice( IPICCTG )
TCpipservice : IPICCTG
GRoup       : IDPROP
DEscription :
Urm        : DFHISAIP
PORtnumber  : 50889           1-65535
STatus     : Open           Open | Closed
PROtocol   : IPic          IIop | Http | Eci | User | IPic
TRansaction : CISS
Backlog    : 00001         0-32767
TSqprefix  :
Host       : ANY
(Mixed Case) :
Ipaddress  : ANY
SOcketclose : No           No | 0-240000 (HHMMSS)
Maxdatalen : 000032       3-524288
SECURITY
SSL        : No           Yes | No | Clientauth
CErtificate :
(Mixed Case) :
PRivacy    :             Notsupported | Required | Supported
CIphers    :
AUTHenticate :           No | Basic | Certificate | AUTOREGISTER
              | AUTOMATIC | ASSERTED
Realm      :
(Mixed Case) :
ATtachsec  :             Local | Verify
DNS CONNECTION BALANCING
DNsgroup   :
GRPcritical : No         No | Yes
  
```

Figure 8-5 CICS TCPIP SERVICE resource definition

2. Configure an IPCONN as shown in Figure 8-6 on page 90:
 - The Tcpiplib should refer to the TCpipservice name created above.
 - The SENDCOUNT should be zero.
 - The APPLID and NETWORKID should match the values specified in the CICS TG PRODUCT section. (Note that they do not reflect actual VTAM applids or network IDs.)
 - LINKAUTH should specify SECUSER.

- SECurityname should specify a user ID suitable to represent this link:
 - It does not need to be your default user ID.
 - It will need READ access to transaction CSMI.
- Userauth should specify IDENTIFY.
- IDprop parameter is not relevant for inbound IPCONN, and so can be left as the default.
- Xlnaction is not required.

```

CEDA View Ipconn( CTG      )
Ipconn      : CTG
Group       : IDPROP
DEscription : RESIDENCY OZ02 IDENTITY PROPOGATION FOR CTG
IPIC CONNECTION IDENTIFIERS
APplid      : MYAPPL
Networkid   : MYNETID
Host        :
(Mixed Case) :
Port        : No                No | 1-65535
Tcpiptime   : IPICCTG
IPIC CONNECTION PROPERTIES
Receivecount : 100              1-999
SEndcount    : 000             0-999
Queuelimit   : No              No | 0-9999
Maxqtime     : No              No | 0-9999
OPERATIONAL PROPERTIES
AUtoconnect  : No              No | Yes
INservice    : Yes            Yes | No
SECURITY
SSl          : No              No | Yes
CErtificate   :                               (Mixed Case)
CIphers       :
Linkauth     : Secuser        Secuser | Certuser
SECurityname  : CICSUSER
Userauth     : Identify      Local | Identify | Verify | Defaultuser
IDprop       : Optional     Notallowed | Optional | Required
RECOVERY
Xlnaction    : Keep            Keep | Force

```

Figure 8-6 CICS IPCONN resource definition

8.4 Configuring identity propagation on WebSphere Application Server

WebSphere application server can be used to pass a distributed identity to a z/OS subsystem such as CICS using CICS Transaction Gateway. See 3.1, “CICS Transaction Gateway” on page 26, for additional details.

To enable support for identity propagation in WebSphere:

1. WebSphere application server must be configured to specify a user registry entry to enable user ID and password verification for applications.
2. All J2EE applications that call the CICS Transaction Gateway ECI resource adapter must be configured for container-managed security.
3. The CICS Transaction Gateway login module (a JAAS module found in the ECI resource adapter RARs - cicseci.rar and cicseciXA.rar) must be installed.
4. One of the following must be configured to use the CICS Transaction Gateway identity propagation login module:
 - The J2EE application must be configured to use a custom login configuration that refers to the CICS Transaction Gateway identity propagation login module. This is accessed via the connection factory resource references on the application's configuration panel.
 - The connection factory that is used by the application must have a mapping configuration alias that refers to the CICS Transaction Gateway identity propagation login module. This is accessed by the connection factory's configuration panel.
5. To propagate the identity of the user who invokes the application, set the `proplIdentity` custom property on the CICS Transaction Gateway identity propagation module to `proplIdentity=Caller`.

The following sections steps describe how to:

- ▶ Configure a standalone LDAP registry.
- ▶ Configure the CICS ECI resource adapter for use with CICS Transaction Gateway.
- ▶ Create a J2C connection factory.
- ▶ Deploy the sample `ECIDateTime` application.
- ▶ Install the identity propagation login module.
- ▶ Run the `ECIDateTime` application.

8.4.1 Configuring standalone LDAP registry

This section provides the steps to configure a standalone LDAP registry for administrative and application security for the identity propagation scenario. For additional information about how to set up administrative and application security, see *WebSphere Application Server V7.0 Security Guide*, SG24-7660:

<http://www.redbooks.ibm.com/abstracts/sg247660.html?Open>

From the WebSphere Application Server administrative console menu go to **Security** → **Global security** and enable administrative security. Do this by checking the Enable administrative security check box (Figure 8-7).



Figure 8-7 Enabling administrative security

Click **Security Configuration Wizard** to perform the configuration for the standalone LDAP registry.

Figure 8-8 shows the first step of the Security Configuration Wizard. Enable application security by checking the box.

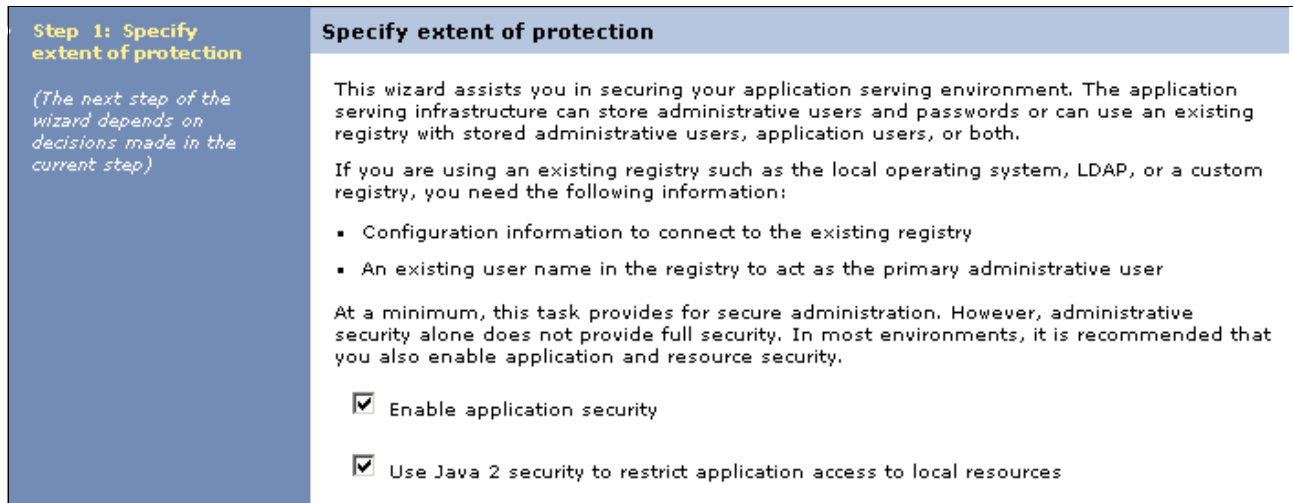


Figure 8-8 Security Configuration Wizard - Step 1

In the next step choose the kind of user repository that you want to use. In this scenario a standalone LDAP registry is used (Figure 8-9).

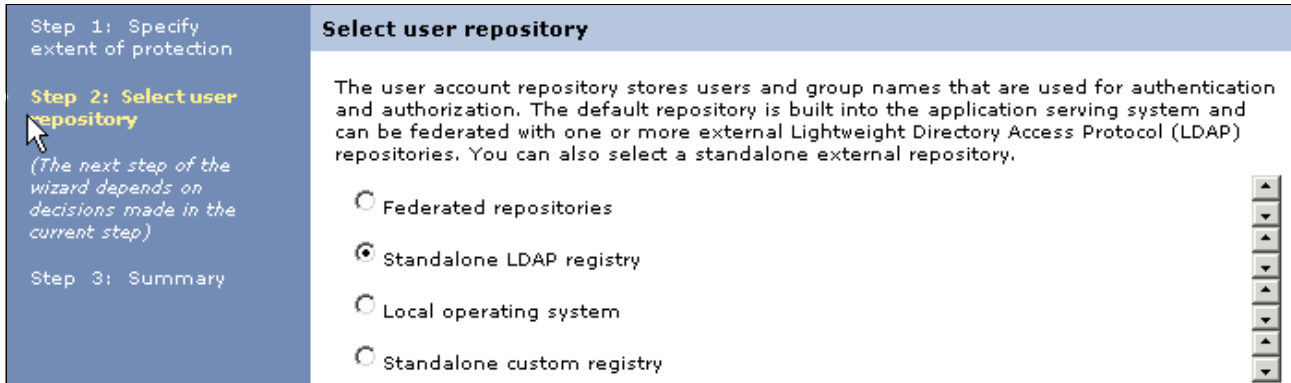


Figure 8-9 Security Configuration Wizard - Step 2

In the third step of the configuration wizard (Figure 8-10) provide a valid primary administrative user name for the LDAP server that you are using. The primary administrative user is a member of the chosen repository, but it also has the same privileges that are associated with the administrative role ID in WebSphere Application Server, and it can access all of the protected administrative methods.

Select the directory type from the Type of LDAP server drop-down menu and provide a host and the port number.

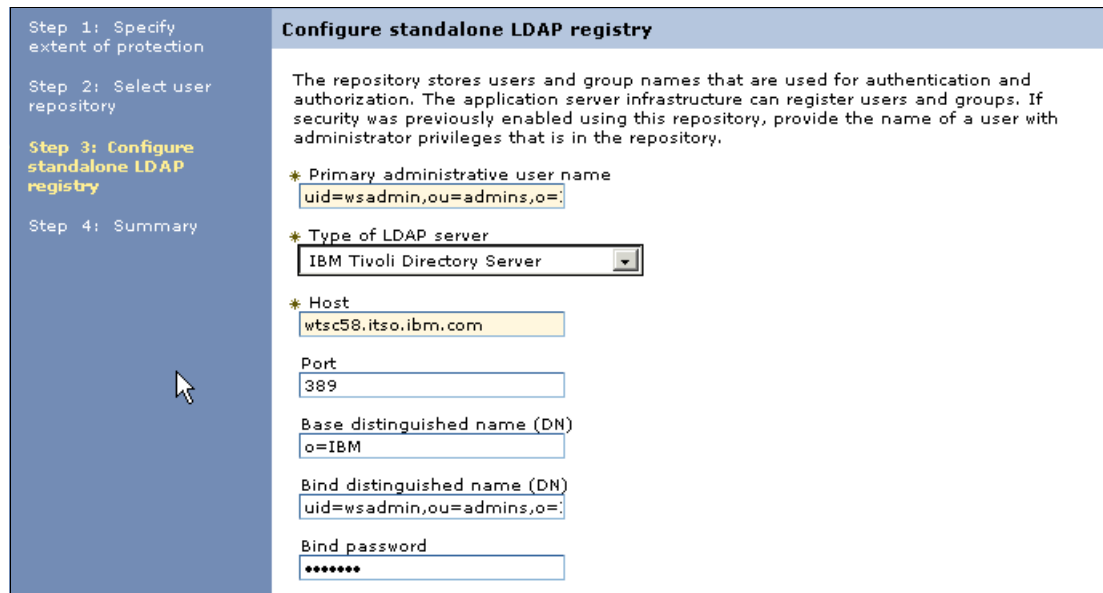


Figure 8-10 Security Configuration Wizard - Step 3

Validate the completed security configuration by clicking **OK** or **Apply**. If there are no validation problems, click **Save** to save the settings to a file that the server uses when it restarts.

The type of LDAP server chosen from the drop-down list in the previous step changes the default object classes that are populated. In this example, IBM Tivoli Directory Server is selected. By selecting this directory type, the default object classes for this directory are populated into the LDAP query strings.

From the Global Security panel select **Configure** on the User account repository section (Figure 8-11).

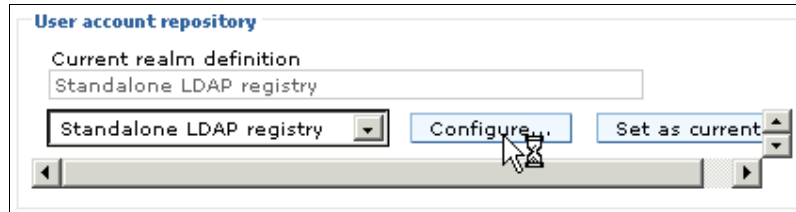


Figure 8-11 Configure user account repository

In the Additional Properties section of the next window, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**. This navigation takes you to the panel shown in Figure 8-12.

Typically, you will be required to modify the object classes specified for the user and group filters. Subsequently, changes to the user ID and group ID mappings might also be needed depending on your directory configuration. These filters and ID mappings change how the application server queries the LDAP directory.

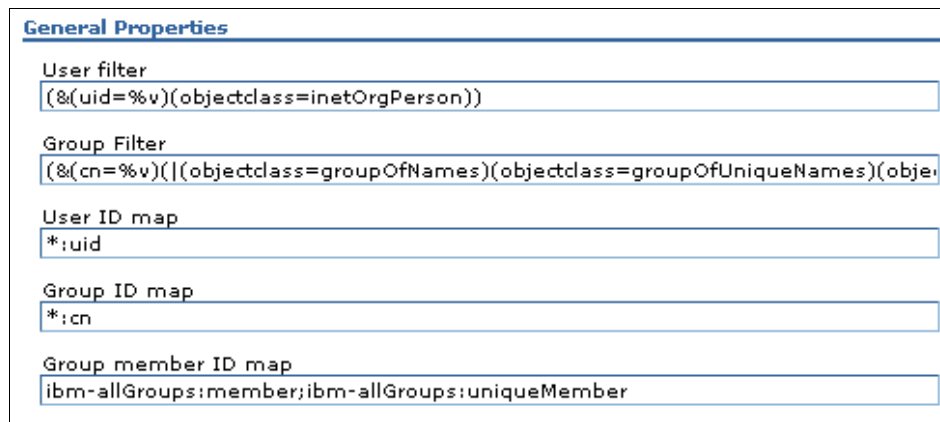


Figure 8-12 Advanced LDAP user registry settings

In this scenario the LDAP directory is using an object class of inetOrgPerson for the users. So, the default object class of ePerson needs to be modified. Change the user filter as shown in Figure 8-12. Click **OK** and save the configuration

Note: You must restart the server for these changes to take effect.

8.4.2 Deploying the CICS ECI resource adapter

The CICS ECI resource adapter is shipped with the CICS Transaction Gateway. It is located under:

```
<install_path>/deployable/cicseci.rar
```

For example, for the test installation:

```
/usr/lpp/cicstg/ctg800/deployable/cicseci.rar
```

From the WebSphere Application Server administrative console menu go to **Resources** → **Resource Adapters** → **Resource Adapters** and click **Install RAR**.

Point to the **cicseci.rar** file (Figure 8-13).

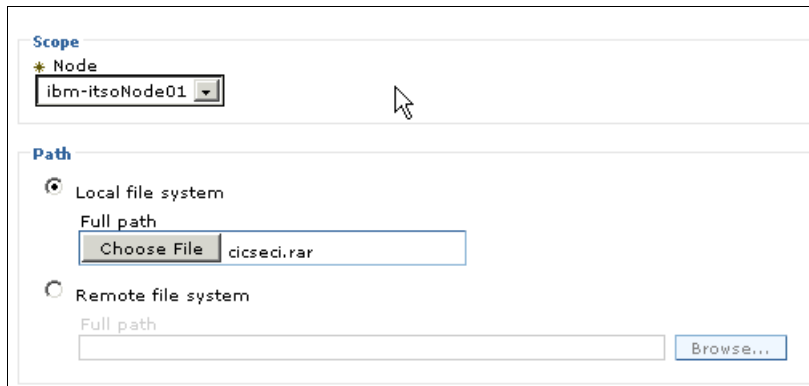


Figure 8-13 Installing CICS ECI adapter

Review the properties (Figure 8-14) and click **OK**.

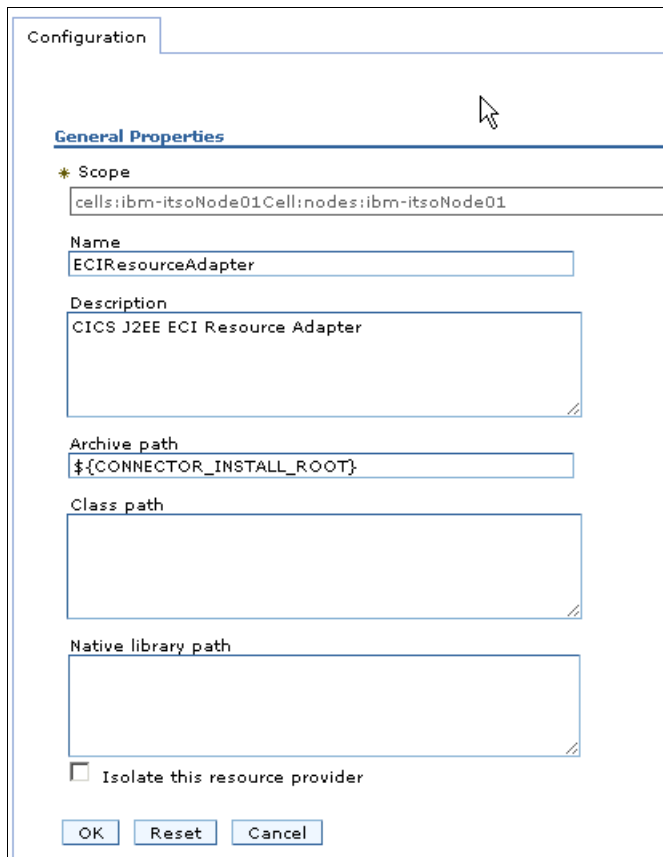


Figure 8-14 General resource adapter properties

You should now be able to see the `ECIResourceAdapter` on the Resource Adapters panel (Figure 8-15).

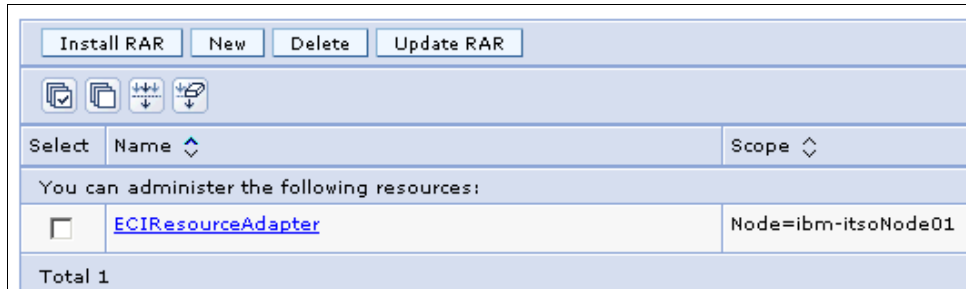


Figure 8-15 `ECIResourceAdapter`

8.4.3 Creating a J2C Connection Factory

From the WebSphere Application Server administrative console menu go to **Resources** → **Resource Adapters** → **J2C connection factories** and click **New**.

Provide a suitable name for the connection factory. Select **`ECIResourceAdapter`** from the drop-down menu. The connection factory must have a JNDI name of `ECI` for the sample program to work (Figure 8-16).

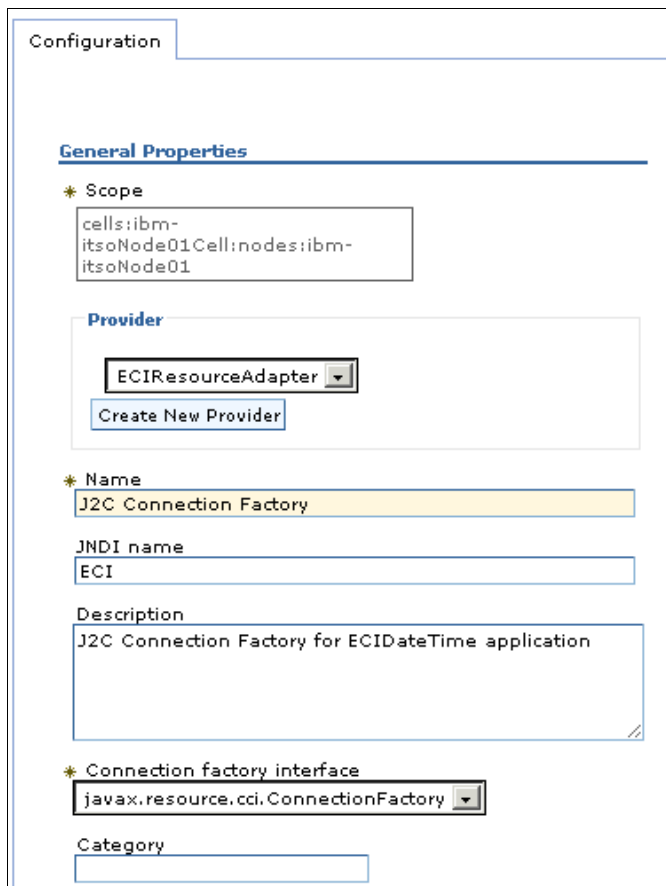


Figure 8-16 J2C connection factory general properties

For now, leave the security settings as is (that is, none).

Return to the J2C Connection factories panel and click the connection factory that you just created (Figure 8-17).

Name	JNDI name	Scope	Provider	Description	Connection factory interface	Category
J2C Connection Factory	ECI	Node=ibm-itsoNode01	ECIResourceAdapter	J2C connection factory for ECIDateTime application	javax.resource.cci.ConnectionFactory	

Figure 8-17 J2C Connection factories

Under Additional properties, click **Custom properties** and provide a ConnectionURL matching your CICS environment. This is the URL of the CICS Transaction Gateway with which the resource adapter will communicate. The URL takes the form protocol://address. This parameter is required. The protocols supported are:

- ▶ TCP
- ▶ SSL
- ▶ Local

In this scenario the connection URL is tcp://wtsc58.itso.ibm.com (Figure 8-18).

Configuration

General Properties

* Scope
cells:ibm-itsoNode01Cell:nodes:ibm-itsoNode01

Required

Name
ConnectionURL

Value
tcp://wtsc58.itso.ibm.com

Description
The URL of the CICS Transaction Gateway for this connection

Type
java.lang.String

Apply OK Reset Cancel

Figure 8-18 Connection URL

Figure 8-19 shows all custom properties that can be set for the ECIResourceAdapter J2C connection factory. For further information refer to the CICS Transaction Gateway InfoCenter, “Deployment parameters for the ECI resource adapters”:

<http://publib.boulder.ibm.com/infocenter/cicstgzo/v8r0/index.jsp?topic=/com.ibm.cics.tg.zos.doc/ctgzos/ccl206.html>

You can administer the following resources:			
TPNName		The transaction identifier of the CICS mirror transaction	false
Applid		The APPLID for applications using this connection	false
ApplidQualifier		The APPLID qualifier for applications using this connection	false
CipherSuites		The cipher suites available for an SSL connection	false
ClientSecurity		The class name of the client security exit for this connection	false
ConnectionURL	tcp://wtsc58.itso.ibm.com	The URL of the CICS Transaction Gateway for this connection	false
KeyRingClass		The location of the keystore containing the certificates required for an SSL connection	false
KeyRingPassword		The password required to access the keystore for an SSL connection	false
Password		The default password that requests through this connection use	false
PortNumber	2006	The port number of the CICS Transaction Gateway for this connection	false
RequestExits		The class name of the request exits called during the execution of interactions	false
ServerName		The name of the target CICS server for this connection	false
ServerSecurity		The class name of the server security exit for this connection requires the Gateway daemon to use	false
SocketConnectTimeout	0	The number of seconds to wait while connecting to a Gateway daemon	false
TraceLevel	1	The level CICS Transaction Gateway diagnostic trace detail	false
TranName		The transaction identifier placed in EIBTRNID by CICS for the mirror transaction	false
UserName		The default user name that requests through this connection use	false

Figure 8-19 custom properties

8.4.4 Deploying the ECIDateTime application

The ECIDateTime application is shipped as a sample CICS Transaction Gateway. It is located under:

```
<install_path>/deployable/ECIDateTime.ear
```

For example, for the test installation:

```
/usr/lpp/cicstg/ctg800/deployable/ECIDateTime.ear
```


From the WebSphere Application Server administrative console select **Applications** → **New Application** → **New Enterprise Application**.

Point to the **ECIDateTime.ear** file (Figure 8-20).

Figure 8-20 Deploy ECIDateTime application

Select **Detailed installation** from the next panel. Use the default values for steps 1 to 4. In step 5, “Provide JNDI names for beans,” validate that the Target Resource JNDI name is set to ECIDateTimeBean1. The ECIDateTimeClient requires this name to be set. ECIDateTimeClient is used for testing in this scenario (Figure 8-21).

EJB module	EJB	URI	Target Resource JNDI Name
ECIDateTimeEJB	ECIDateTime	ECIDateTimeEJB.jar,META-INF/ejb-jar.xml	ECIDateTimeBean1

Figure 8-21 Deploy ECIDateTime application - Step 5

In step 6, “Map resource references to resources,” select **ECIDateTimeEJB** and click **Browse** for Target Resource JNDI Name. Select the J2C Connection Factory that you created earlier (Figure 8-22) and click **Apply**.

Select	Name	JNDI name	Scope	Description
<input checked="" type="radio"/>	J2C Connection Factory	ECI	Node=ibm-itsoNode01	J2C connection factory for ECIDateTime applica

Figure 8-22 Map resource references to resources

As a result, the JNDI name ECI is set under the Target Resource JNDI Name column (Figure 8-23).

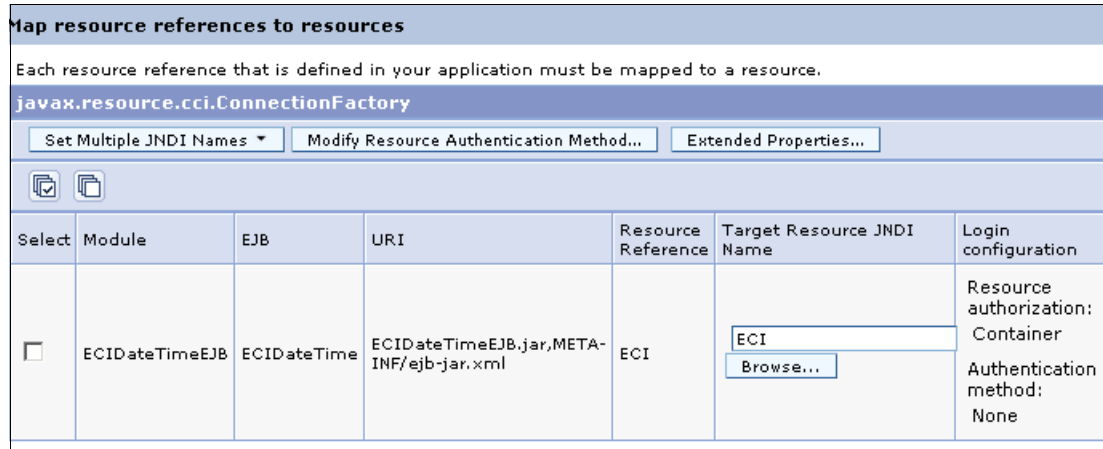


Figure 8-23 Deploy ECIDateTime application - Step 6

8.4.5 Installing the identity propagation login module

Ensure that application security is enabled from the Global Security panel, then select **Java Authentication and Authorization Service** → **Application logins** and click **New**.

Enter CTG_idprop as the alias for the new login module (Figure 8-24) and click **OK**.

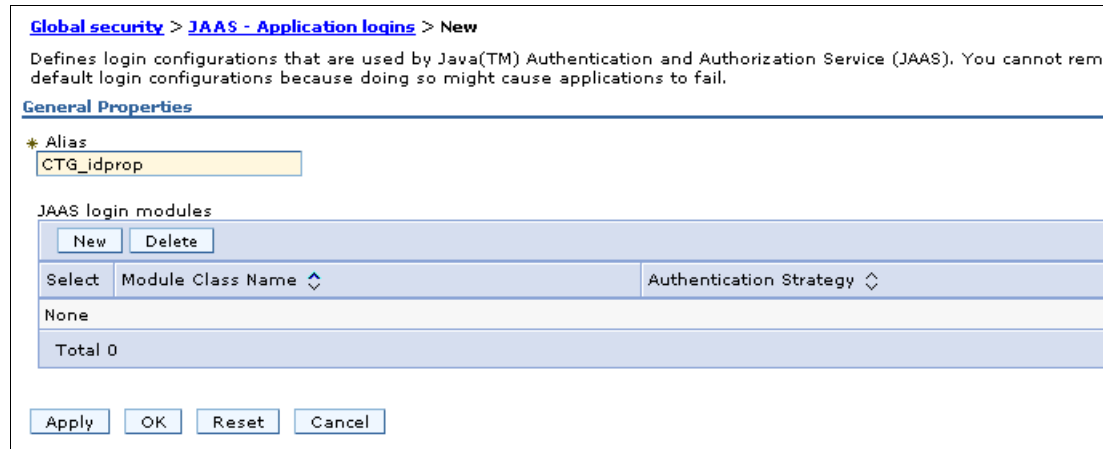


Figure 8-24 New JAAS application login

Return to the previous panel and click the **CTG_idprop** login module. Click **New** and set the module class name to `com.ibm.ctg.security.idprop.LoginModule`. Select **REQUIRED** from the Authentication strategy drop-down menu, create the name-value pair `propIdentity-RunAs` (Figure 8-25), and click **OK**.

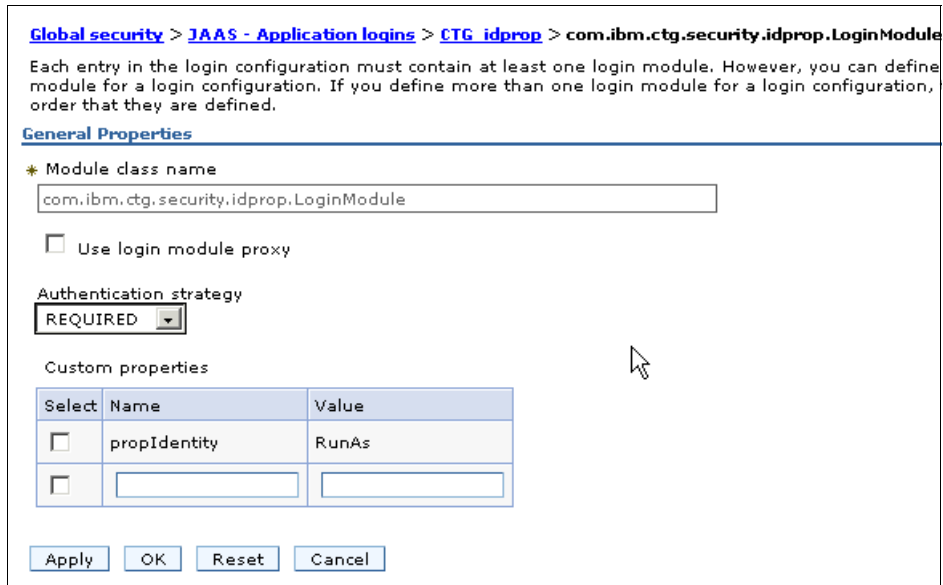


Figure 8-25 CTG_idprop properties

To associate the identity propagation login module with the client application go to **Applications** → **Application types** → **WebSphere enterprise applications** and select the **ECIDateTime** application from the list.

Select **Resource references**.

Select the ECI resource reference using the check box, and then select **Modify Resource Authentication Method** (Figure 8-26).

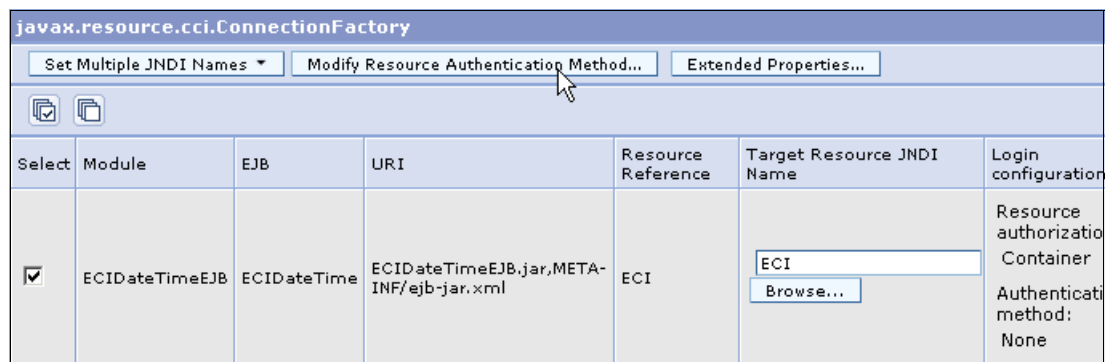


Figure 8-26 Modify Resource Authentication Method

Select **Use custom login configuration** and then select the identity propagation login module **CTG_idprop** that you installed in the previous step from the drop-down menu and click **Apply** (Figure 8-27).

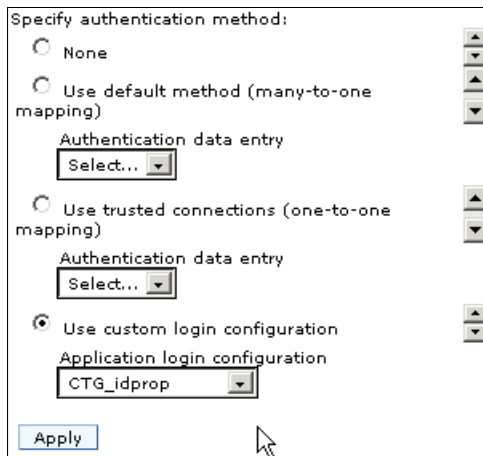


Figure 8-27 Custom login configuration

Save the configuration changes.

From the administrative console select **Applications** → **Application Types** → **WebSphere enterprise applications**.

Select **ECIDateTime** and click **Start** (Figure 8-28).

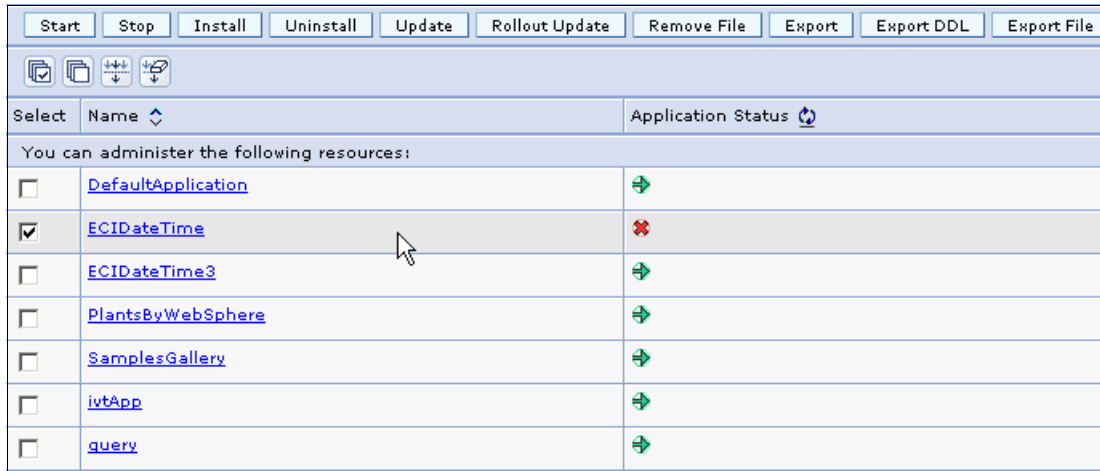


Figure 8-28 Start ECIDateTime

8.4.6 Running the ECIDateTime application

From the Windows command prompt navigate to `<path>\profiles\<appserver_name>\bin` and start the launchClient utility by issuing the following command:

```
launchClient <filepath>\ECIDateTime.ear
```

Where *filepath* is the path to the ECIDateTime.ear file. The command prompt for the installed scenario would be:

```
launchClient ..\installedApps\ibm-itsoNode01Ce11\ECIDateTime.ear
```

When application security is enabled, a dialog asks for the credentials (username and password). Because the application is being authenticated against an LDAP registry, you have to supply the distinguished name that has been defined in the LDAP registry (uid=martina,ou=swg,o=ibm). The password is also required (Figure 8-29).



Figure 8-29 ECIDateTime login credentials

You now see the return code ECI_ERR_SECURITY_ERROR and a Java stack trace in the console. The exception starts as shown in Example 8-1.

Example 8-1 ECIDateTime application output

```

javax.resource.spi.SecurityException: CTG9631E Error occurred during interaction
with CICS:
ECI_ERR_SECURITY_ERROR, error code: -27

```

See CICS user message log and the JES message log for the CICS job. If there is an error in your configuration you will not see the distributed user ID in the logs.

You can confirm connectivity between CICS TG and CICS by observing the messages shown in Example 8-2 in CICS and Example 8-3 in CICS TG.

Example 8-2 CICS indicating IPCONN successful communication with CICS TG

```

DFHIS2001 02/01/2011 11:18:36 SC58CIC1 Client web session 1 from applid MYAPPL
accepted for IPCONN CTG.
DFHIS2001 02/01/2011 11:18:36 SC58CIC1 Client web session 2 from applid MYAPPL
accepted for IPCONN CTG.

```

Example 8-3 CICS TG indicating IPIC server successful communication with CICS TG

```

02/01/11 11:18:36:884 Y0" CTG6506I Client connected: YConnectionManager-99" -
tcp@SocketYaddr=/9.12.5.202,port=4659,localport=2006"
02/01/11 11:18:36:907 Y0" CTG8429I Established new IPIC connection to CICS server
ITSOLAB with: negotiated session limit=100, CICSAP
PLID=SC58CIC1 CICSAPPLIDQUALIFIER=USIBMSC , HOSTNAME=wtsc58.itso.ibm.com,
PORT=50889, sockets=2

```

For your initial test, the CICS user message log will contain the message shown in Example 8-4.

Example 8-4 CICS MSGUSR: Configuration error

```

DFHIS1027 01/27/2011 09:49:19 SC58CIC1 Security violation has been detected using
IPCONN CTG and transaction id CSMI by userid ????????

```

Consider installing the CICS SupportPac CH51, which allows for more detailed monitoring by providing output to a JES DD log, //TXNLOG, that allows you to verify receipt of a distributed identity in CICS TG from WebSphere Application Server.

In the CICS TG TXNLOG in Example 8-5 there is no distributed identity received in CICS TG from WebSphere Application Server, which confirms a WebSphere Application Server setup error.

Example 8-5 CTG TXNLOG: Configuration error

```
01/27/11 09:49:19:119 CH51 -> Rqst(371) Flow(ExtendedModeEci) Pgm(EC01)
ClientIP(9.12.5.202) commarea(20)bytes SocketData(131)bytes LwToken(0)
01/27/11 09:49:19:142 CH51 <- Rqst(371) Flow(ExtendedModeEci) Srv(null/ITSOLAB)
Pgm(EC01) ClientIP(9.12.5.202) commarea(20)bytes SocketData(83)bytes
RespTime(23)ms CICScall(0)ms ctg_rc(-27)
TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C73E556A8DCA9E00) LwToken(0)
```

After you have WebSphere Application Server configured to provide application security, you can then see the distributed identity flow through CICS TG. At this point no mappings have been defined to RACF, so a security failure is expected, as you can see in Example 8-6, Example 8-7, and Example 8-8.

Example 8-6 CICS TG TXNLOG showing DistId flowing

```
01/27/11 10:06:16:416 CH51 -> Rqst(382) Flow(ExtendedModeEci) Pgm(EC01)
DistID(wtsc58.itso.ibm.com:389/uid=martina,ou=swg,o=ibm) ClientIP(9.12.5.202)
commarea(20)bytes SocketData(183)bytes LwToken(0)
01/27/11 10:06:16:420 CH51 <- Rqst(382) Flow(ExtendedModeEci) Srv(null/ITSOLAB)
Pgm(EC01) DistID(wtsc58.itso.ibm.com:389/uid=martina,ou=swg,o=ibm)
ClientIP(9.12.5.202) commarea(20)bytes SocketData(83)bytes RespTime(4)ms
CICScall(2)ms ctg_rc(-27)
TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C73E5934B46F2000) LwToken(0)
```

Example 8-7 CICS MSGUSR: No mapping

```
DFHIS1027 01/27/2011 10:06:16 SC58CIC1 Security violation has been detected using
IPCONN CTG and transaction id CSMI by userid ????????
```

The JES message log for the CICS job contains the message shown in Example 8-8.

Example 8-8 CICS JESMSG LG: No mapping

```
10.06.16 STC03863 ICH408I USER(STC ) GROUP(TSO ) NAME(STARTED TASK)
646 DISTRIBUTED IDENTITY IS NOT DEFINED:
646 uid=martina,ou=swg,o=ibm wtsc58.itso.ibm.com:389
10.06.16 STC03863 IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.
```

In addition, you might want to see the transaction log of the CICS Transaction Gateway. This confirms the flowing of the distributed identity (Example 8-9).

Example 8-9 CICS TG TXNLOG: No mapping

```
01/27/11 10:06:16:416 CH51 -> Rqst(382) Flow(ExtendedModeEci) Pgm(EC01)
DistID(wtsc58.itso.ibm.com:389/uid=martina,ou=swg,o=ibm) ClientIP(9.12.5.202)
commarea(20)bytes SocketData(183)bytes LwToken(0)
01/27/11 10:06:16:420 CH51 <- Rqst(382) Flow(ExtendedModeEci) Srv(null/ITSOLAB)
Pgm(EC01) DistID(wtsc58.itso.ibm.com:389/uid=martina,ou=swg,o=ibm)
```

```
ClientIP(9.12.5.202) commarea(20)bytes SocketData(83)bytes Resp Time(4)ms
CICScall(2)ms ctg_rc(-27)
TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C73E5934B46F2000) LwToken(0)
```

The SMF records related to this contain the information given in Example 8-10.

Example 8-10 SMF output: No mapping

```
S M F   R E C O R D   L I S T I N G   19Jan11 11:43 to 19Jan11 18:02

Date      Time      Typ User      Event      Eq
19Jan2011 16:42:52  80 STC      RACINIT    39
                               uid=martina,ou=swg,o=ibm
                               wtsc58.itso.ibm.com:389
```

This example is formatted using a zSecure Consul Auditing and Reporting Language (CARLa) script. CARLa is the main reporting engine used within zSecure Admin, zSecure Audit, zSecure Alert, zSecure Visual, and zSecure Manager for RACF z/VM. The event qualifier (Eq) of 39 describes a logon violation due to no mapping.

If these messages appear, this is not an indication of a problem at this stage. The messages are expected because, although the connection to CICS was established, the application failed because the LDAP identity was not mapped to a RACF user ID. In the next step in 8.5, “Configuring identity propagation on z/OS” on page 105, the propagation will be configured.

Note: We did find the DFHIS1027 message to be confusing. It can be generated both for “No distributed Identity provided” and also for “No mapping from RACF” situations. It also suggests a RACF access failure using terminology *security violation* when, in fact, no RACF access check has occurred. You are unable to distinguish between these two situations from the message alone.

- ▶ With PTF UK63945 applied, the message does show a user ID of eight question marks (that is, ?????????). It has no identity available. Previously, CICS was reporting the SECURITYNAME value from the IPCONN definition.
- ▶ In the “No mapping found from RACF” situation, you will see a ICH408I message in the JES log indicating an unmapped distributed identity, and an SMF record for RACF event 1, EventQualifier 39.

8.5 Configuring identity propagation on z/OS

This work might have already been performed by your z/OS systems programmer. However, we found that the z/OS 1.11 installation documentation did not indicate that it is a required action, so it may have been omitted.

1. Update the IKJTSOxx member of SYS1.PARMLIB to add the new RACMAP command to the AUTHCMD section.
2. Update z/OS either by using the PARMLIB command or IPL.

8.6 Configuring identity propagation on RACF

To configure identity propagation on RACF:

1. Activate IDIDMAP class.
2. Issue suitable RACMAP commands that will map a distributed identity to a RACF user ID. See Chapter 4, “RACMAP function” on page 39, for details on RACMAP. See Chapter 5, “Filter management” on page 49, for more details on establishing a mapping philosophy regarding one-to-one” and many-to-one mappings.

In this example we use two mappings, one for ‘uid=martina,ou=swg,o=ibm’ and a generic one for anyone else in ‘ou=swg, o=ibm’.

- ADDUSER SWGDE
- ADDUSER SWGRES
- RACMAP ID(SWGDE) USERDIDFILTER(name('uid=martina,ou=swg,o=ibm'))
REGISTRY(name('wtsc58.itso.ibm.com:389'))
- RACMAP ID(SWGRES) USERDIDFILTER(name('ou=swg,o=ibm'))
REGISTRY(name('wtsc58.itso.ibm.com:389'))
- SETROPTS RACLIST(IDIDMAP) REFRESH

3. For the purpose of this testing, we wanted to ensure that we would see the RACF audit records in SMF, so we audited all uses of transaction CSMI.
 - RALTER TCICSTRN CSMI AUDIT(ALL(READ))
 - SETROPTS RACLIST(TCICSTRN) REFRESH

Warning: We do *not* expect you to audit all uses of CSMI in a production environment. That could degrade system performance. Audit activity based on your own requirements.

8.7 Testing the scenario

The following tests were performed:

- ▶ No distributed identity passed.
- ▶ Distributed identity passed, but no mapping exists for that identity.
- ▶ Mapping exists via a one-to-one mapping.
- ▶ Mapping exists via a many-to-one mapping.
- ▶ Testing of DPL call across an MRO link to a second CICS region showing a program using CICS API INQUIRE ASSOCIATION.

Repeat testing after RACF mappings have changed. Each test consists of running a simple Date/Time WebSphere Application Server application that drives a DPL to a connected region and that in turn drives a DPL link to a second CICS region.

Information from the JES log, CICS TG monitor log, CICS TD queue CSSL (//MSGUSR), SMF reporting, and program reporting (also to CSSL) are shown below.

8.7.1 Results for No Distributed Identity Passed

This section shows the results of various logs when no distributed identity has been passed.

JES log

Nothing is recorded.

CICS TG/TXNLOG log (with CICS SupportPac CH51 applied)

Example 8-11 shows the results from the CICS TG TXNLOG show no DistID information.

Example 8-11 CICS TG TXNLOG: No distributed identity flowed from WebSphere Application Server into CICS TG

```
01/27/11 09:49:19:119 CH51 -> Rqst(371) Flow(ExtendedModeEci) Pgm(EC01)
ClientIP(9.12.5.202) commarea(20)bytes SocketData(131)bytes LwToken(0)
01/27/11 09:49:19:142 CH51 <- Rqst(371) Flow(ExtendedModeEci) Srv(null/ITSOLAB)
Pgm(EC01) ClientIP(9.12.5.202) commarea(20)bytes SocketData(83)bytes
RespTime(23)ms CICScall(0)ms ctg_rc(-27)
TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C73E556A8DCA9E00) LwToken(0)
```

CICS CSSL/MSGUSR log

The CICS CSSL MSGUSR log records a security violation when distributed identity is passed (Example 8-12).

Example 8-12 CICS MSGUSR: No distributed identity flowed into CICS

```
DFHIS1027 01/27/2011 09:49:19 SC58CIC1 Security violation has been detected using
IPCNN CTG and transaction id CSMI by userid ????????
```

RACF auditing to SMF

Nothing is recorded.

8.7.2 Results for No mapping found

This section shows the results of various logs when no matching filter has been found for a distributed identity in RACF.

JES log

The CICS JESMSGGLG records the ICH408I message indicating that no matching filter has been found for the distributed identity (Example 8-13).

Example 8-13 CICS JESMSGGLG: CICS passed distributed identity to RACF, but no mappings existed

```
10.06.16 STC03863 ICH408I USER(STC ) GROUP(TSO ) NAME(STARTED TASK)
646 DISTRIBUTED IDENTITY IS NOT DEFINED:
646 uid=martina,ou=swg,o=ibm wtsc58.itso.ibm.com:389
10.06.16 STC03863 IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.
```

CICS TG/TXNLOG log (with CICS SupportPac CH51 applied)

The CICS TG TXNLOG shows that a distributed identity has been passed, as indicated by the DistID information (Example 8-14).

Example 8-14 CICS TG TXNLOG showing DistId flowing

```
01/27/11 10:06:16:416 CH51 -> Rqst(382) Flow(ExtendedModeEci) Pgm(EC01)
DistID(wtsc58.itso.ibm.com:389/uid=martina,ou=swg,o=ibm) ClientIP(9.12.5.202)
commarea(20)bytes SocketData(183)bytes LwToken(0)
01/27/11 10:06:16:420 CH51 <- Rqst(382) Flow(ExtendedModeEci) Srv(null/ITSOLAB)
Pgm(EC01) DistID(wtsc58.itso.ibm.com:389/uid=martina,ou=swg,o=ibm)
ClientIP(9.12.5.202) commarea(20)bytes SocketData(83)bytes RespTime(4)ms
CICScall(2)ms ctg_rc(-27)
TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C73E5934B46F2000) LwToken(0)
```

CICS CSSL/MSGUSR log

The CICS CSSL MSGUSR log records a security violation when no match is found in RACF for a distributed identity (Example 8-15).

Example 8-15 CICS MSGUSR: Distributed identity flowed into CICS, but RACF had no mapping

```
DFHIS1027 01/27/2011 10:06:16 SC58CIC1 Security violation has been detected using
IPCINN CTG and transaction id CSMI by userid ????????
```

RACF auditing to SMF

The SMF record shows a RACINIT event for the attempted mapping of the distributed identity (Example 8-16).

Example 8-16 SMF report of no mapping, RACF Event 1, Qualifier 39, showing distributed identity

```
S M F   R E C O R D   L I S T I N G   27Jan11 09:30:43 to 27Jan11 11:32

Date      Time      Typ User      Event      Eq
27Jan2011 10:06:16  80 STC      RACINIT    39
                uid=martina,ou=swg,o=ibm
                wtsc58.itso.ibm.com:389
```

8.7.3 Results for one-to-one mapping

This test shows that a one-to-one mapping occurs and that the SMF audit trail shows the distributed identity.

The mapping that was in effect at this time was USERDIDFILTER(name('UID=MARTINA,OU=SWG,O=IBM')), which mapped to user ID SWGDE.

JES log

Nothing is recorded.

CICS TG/TXNLOG log (with CICS SupportPac CH51 applied)

The CICS TG TXNLOG shows that a distributed identity has been passed, as indicated by the DistID information (Example 8-17).

Example 8-17 Sample CICS TG log for one-to-one mapping

```
02/01/11 12:25:04:208 CH51 -> Rqst(508) Flow(ExtendedModeEci) Pgm(EC01)
DistID(wtsc58.itso.ibm.com:389/UID=MARTINA,OU=SWG,O=IBM) ClientIP(9.12.5.202)
commarea(20)bytes SocketData(183)bytes LuwToken(0)
```

```
02/01/11 12:25:04:875 CH51 <- Rqst(508) Flow(ExtendedModeEci) Srv(null/ITSOLAB)
Pgm(EC01) DistID(wtsc58.itso.ibm.com:389/UID=MARTINA,OU=SWG,O=IBM)
ClientIP(9.12.5.202) commarea(20)bytes SocketData(83)bytes RespTime(667)ms
CICScall(637)ms TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C744C18E09DBC400)
LuwToken(81)
```

```
02/01/11 12:25:04:877 CH51 -> Rqst(509) Flow(ExtendedModeCommit)
ClientIP(9.12.5.202) SocketData(123)bytes LuwToken(81)
```

```
02/01/11 12:25:04:879 CH51 <- Rqst(509) Flow(ExtendedModeCommit) Srv(null/ITSOLAB)
ClientIP(9.12.5.202) SocketData(58)bytes RespTime(2)ms CICScall(1)ms
TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C744C18E09DBC400)
```

CICS CSSL/MSGUSR log

Nothing is recorded.

RACF auditing to SMF

When a distributed identity has been successfully mapped to a RACF user ID, the SMF records will contain both the RACF user ID and the distributed identity for resource access (Example 8-18).

Example 8-18 Sample SMF report showing audited access from both regions

S M F R E C O R D L I S T I N G 1Feb11 11:18 to 1Feb11 12:58							
Date	Time	Jobname	UserId	Event	Eq Class	Intent	Resource
01Feb2011	12:25:04	SC58CIC1	SWGDE	ACCESS	0 TCICSTRN	READ	CSMI
			UID=MARTINA,OU=SWG,O=IBM				wtsc58.itso.ibm.com:389

8.7.4 Results for many-to-one mapping

This test shows that when RACF does not find a one-to-one mapping, it strips off the leading RDN, and then searches again.

The mapping that was in effect at this time was USERDIDFILTER(name('OU=SWG,O=IBM')), which mapped to user ID SWGRES. The key difference between this test and the previous test is that even with the identical DN and realm coming in, we set up the mapping filters in RACF to map to a different user ID. User ID SWGRES is mapped via a many-to-one filter consisting of just the OU and O RDNs.

JES log

Nothing is recorded.

CICS TG/TXNLOG log (with CICS SupportPac CH51 applied)

The CICS TG TXNLOG shows that a distributed identity has been passed, as indicated by the DistID information (Example 8-19).

Example 8-19 CICS TG TXNLOG showing DistId flowing

```
01/19/11 17:32:20:916 CH51 -> Rqst(74) Flow(ExtendedModeEci) Pgm(EC01)
DistID(wtsc58.itso.ibm.com:389/UID=MARTINA,OU=SWG,0=IBM) ClientIP(9.12.5.202)
commarea(20)bytes SocketData(183)bytes LwToken(0)
01/19/11 17:32:20:987 CH51 <- Rqst(74) Flow(ExtendedModeEci) Srv(null/ITSOLAB)
Pgm(EC01) DistID(wtsc58.itso.ibm.com:389/UID=MARTINA,OU=SWG,0=IBM)
ClientIP(9.12.5.202) commarea(20)bytes SocketData(83)bytes RespTime(71)ms
CICScall(44)ms TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C734ADFDA84B2700)
LwToken(17)
01/19/11 17:32:20:992 CH51 -> Rqst(75) Flow(ExtendedModeCommit)
ClientIP(9.12.5.202) SocketData(123)bytes LwToken(17)
01/19/11 17:32:20:995 CH51 <- Rqst(75) Flow(ExtendedModeCommit) Srv(null/ITSOLAB)
ClientIP(9.12.5.202) SocketData(58)bytes RespTime(3)ms CICScall(1)ms
TrnGrpId(180ED4E8D5C5E3C9C42ED4E8C1D7D7D3C734ADFDA84B2700) LwToken(17)
```

CICS CSSL/MSGUSR log

The CICS CSSL MSGUSR log indicates session activity (Example 8-20).

Example 8-20 CICS log showing session establishment

```
DFHIS2001 01/19/2011 17:32:20 SC58CIC1 Client web session 1 from applid MYAPPL
accepted for IPCONN CTGIDPRO.
DFHIS2001 01/19/2011 17:32:20 SC58CIC1 Client web session 2 from applid MYAPPL
accepted for IPCONN CTGIDPRO
```

RACF auditing to SMF

We had AUDIT(ALL(READ)) for transaction CSMI to force auditing for this SMF report (Example 8-21).

Example 8-21 SMF output: Resource access audited after successful mapping

```
S M F   R E C O R D   L I S T I N G   19Jan11 11:43 to 19Jan11 18:02
```

Date	Time	Typ	User	Event	Eq	Class	Intent	Resource
19Jan2011	17:32:20	80	SWGRES	ACCESS	0	TCICSTRN	READ	CSMI
								UID=MARTINA,OU=SWG,0=IBM
								wtsc58.itso.ibm.com:389

8.7.5 Results of DPL to second CICS showing INQUIRE ASSOCIATION

For this test we adjusted WebSphere Application Server to drive program EC03. EC03 was functionally similar to EC01, and had additional code added to retrieve the DN and realm and print these in the MSGUSR log via CSSL. It also drove a link to program EC04 in another CICS region to demonstrate that the distributed identity is propagated between CICS regions, and similarly retrieve the DN and realm and print it.

In this case we used a lowercase signon to LDAP and a lowercase mapping in RACF.

Note: Programs EC03 and EC04 mentioned here are not the CICS samples but our own programs modeled after EC01.

Snippets of COBOL program to retrieve distributed identity

CICS applications can determine the RACF user ID running a transaction by using EXEC CICS ASSIGN USERID(WS-FIELD). With identity propagation, an application can now retrieve the DN and realm via the CICS API call EXEC CICS INQUIRE ASSOCIATION (Example 8-22).

Example 8-22 Sample COBOL code to retrieve user ID and distributed identity

```
05 WS-TASK-NBR          PIC S9(7) COMP-3.
05 WS-USERID            PIC X(8)  VALUE SPACES.
05 WS-DISTID            PIC X(246) VALUE SPACES.
05 WS-DISTREG           PIC X(252) VALUE SPACES.

EXEC CICS ASSIGN USERID(WS-USERID) END-EXEC.
MOVE EIBTASKN TO WS-TASK-NBR.
EXEC CICS INQUIRE ASSOCIATION(WS-TASK-NBR)
                   DNAME(WS-DISTID)
                   REALM(WS-DISTREG)

END-EXEC
```

These fields are returned to you in UTF-8 format, so you might want to translate to EBCDIC. We chose to use a CICS container to perform such translation. You might prefer to use alternative methods. We wrote those working storage fields to TD queue CSSL to show in Example 8-23.

Example 8-23 Sample COBOL code to translate UTF8 to EBCDIC

```
05 EBCDIC              PIC S9(8) BINARY VALUE +37.
05 UTF8                PIC S9(8) BINARY VALUE +1208.
05 WS-LEN246           PIC S9(7) BINARY VALUE +246.
05 WS-DISTID           PIC X(246) VALUE SPACES.

EXEC CICS PUT CONTAINER('TEMP') CHANNEL('TEMP')
             FROM(WS-DISTID) FLENGTH(WS-LEN246)
             DATATYPE(DFHVALUE(CHAR)) FROMCCSID(UTF8)

END-EXEC
EXEC CICS GET CONTAINER('TEMP') CHANNEL('TEMP')
             INTO(WS-DISTID) FLENGTH(WS-LEN246)
             INTOCCSID(EBCDIC)

END-EXEC
```

With this information now available within the CICS application, it can be used for business purposes within that application, such as an internal audit trail, or to drive business logic (Example 8-24).

Example 8-24 Sample display of translated DNAME and REALM from CICS application

```
01/02/11 12:53:34 Program EC03 invoked by userid SWGDE
01/02/11 12:53:34 Remote Identity: uid=martina,ou=swg,o=ibm
01/02/11 12:53:34 Remote Registry: wtsc58.itso.ibm.com:389
```

```
01/02/11 12:53:34 Linking to EC04, CommArea=  
01/02/11 12:53:34 Resp: Normal , CommArea=01/02/11 12:53:34 SC58CIC2
```

To confirm that the distributed identity is propagated to connected CICS regions, we performed a DPL call to program EC04, which repeated the API calls to retrieve the user ID and DN (Example 8-25).

Note: identity propagation will work on either an MRO connection or an IPCONN connection. It will not work on an APPC connection. Our testing used an IPCONN from CICS TG to CICS, and MRO from CICS to CICS.

Example 8-25 Sample display of translated DNAME & REALM from connected CICS region

```
01/02/11 12:53:34 SC58CIC2 Program EC04 invoked by userid SWGDE  
01/02/11 12:53:34 SC58CIC2 Remote Identity: uid=martina,ou=swg,o=ibm  
01/02/11 12:53:34 SC58CIC2 Remote Registry: wtsc58.itso.ibm.com:389
```

SMF report showing distributed identity available from both regions

Example 8-26 shows that the distributed identity is logged in the audit trail from both regions.

Example 8-26 Sample SMF report showing audited access from both regions

```
S M F   R E C O R D   L I S T I N G   1Feb11 11:18 to 1Feb11 12:58  
  
Date      Time      Jobname  UserId    Event     Eq Class  Intent  Resource  
  
01Feb2011 12:53:34 SC58CIC1 SWGDE    ACCESS    0 TCICSTRN READ    CSMI  
uid=martina,ou=swg,o=ibm  
wtsc58.itso.ibm.com:389  
01Feb2011 12:53:34 SC58CIC2 SWGDE    ACCESS    0 TCICSTRN READ    CSMI  
uid=martina,ou=swg,o=ibm  
wtsc58.itso.ibm.com:389
```

8.7.6 Results after RACF mappings have changed

What happens to the cached identity data if RACF mappings change? In our testing of the CICS TG configuration, we observed that new RACMAP rules only took effect after the interval specified in the CICS USRDELAY SIT option had expired.

After the USRDELAY interval had expired, we observed a RACROUTE VERIFY DELETE and a subsequent inbound request from the CICS TG caused a new RACROUTE VERIFY CREATE that used the new mappings. The reason for this behavior is that RACF does not invoke the ENF mechanism when mappings are changed, and so CICS uses the USRDELAY interval to control the amount of time used to store RACF mappings within a CICS region. See 2.3, “Impact on distributed identity data cached in RACF by z/OS subsystems” on page 16, for a discussion of the ENF mechanism.



Identity propagation with DB2 for z/OS

This chapter demonstrates how a distributed identity is passed to DB2 for z/OS using JDBC/DB2 Connect™. We produce audit reports to show details of the distributed user.

The supported configurations are:

- ▶ DB2 10 for z/OS
- ▶ z/OS Version 1 Release 11
- ▶ DB2 Connect V9.7

This example does not configure an LDAP registry. Instead, the distinguished name (DN) and registry name are coded in the program. The expectation is the JAVA developers would interrogate the LDAP server, and not code the distributed identity in the program.

9.1 JAVA application test scenario

A sample JAVA application running on Windows accesses DB2 for z/OS using a distributed identity, which is normally found in a LDAP directory, but in this example is coded in the sample JAVA application. The main purpose for this test is to show the distributed identity association with the DB2 thread.

Figure 9-1 shows the steps taken by the application, such as connecting to DB2 for z/OS and ultimately executing SQL.

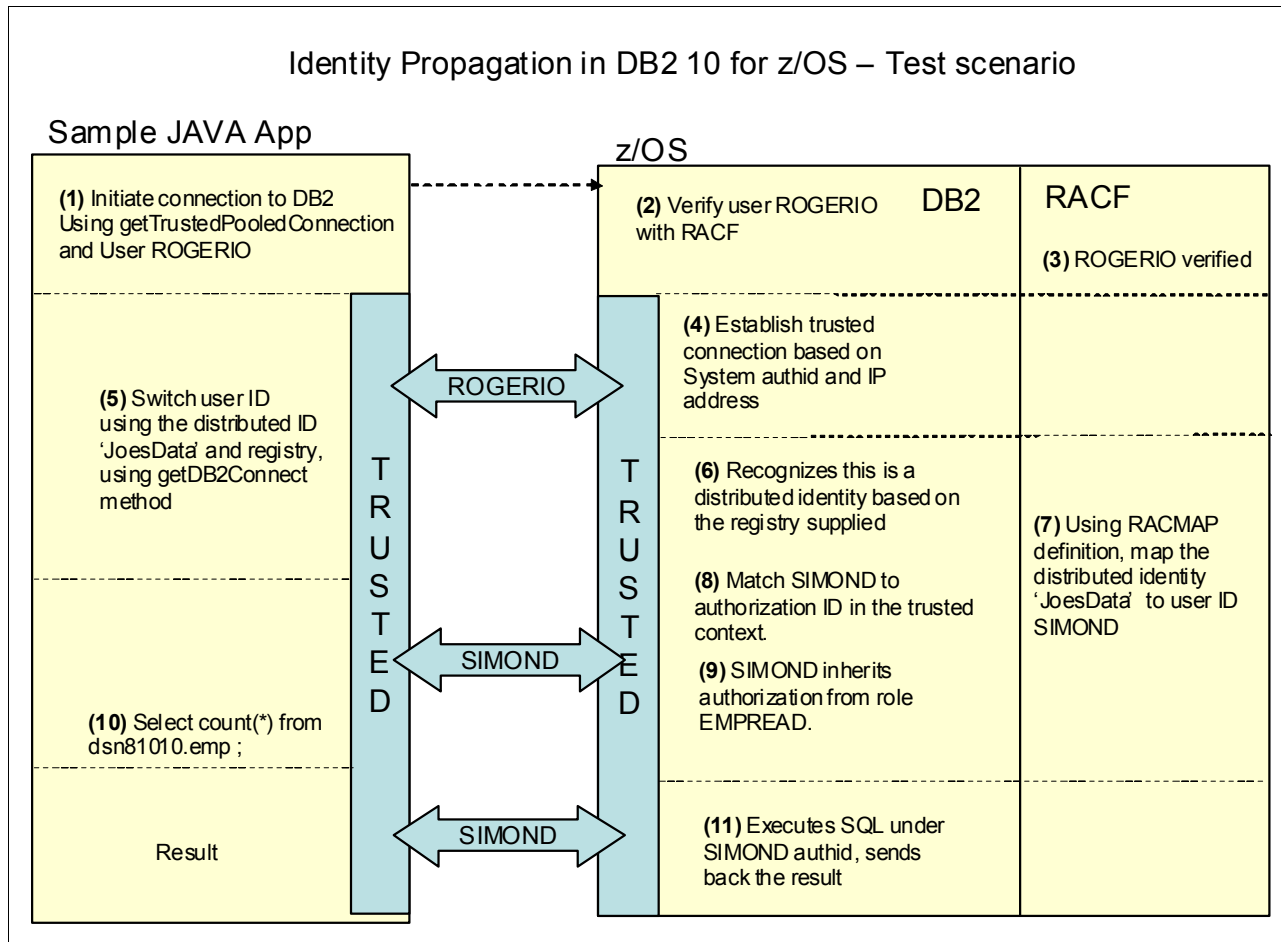


Figure 9-1 Sample JAVA app accessing DB2 for z/OS on a trusted connection

Figure 9-1 displays the process flow, starting with establishing the connection from the JAVA application running on Windows, then having RACF verify the user ID. Using the method `getTrustedPooledConnection`, DB2 establishes the trusted connection based on the system authid and IP address. Using the `getDB2Connect` method, the distributed user ID and registry name is passed to DB2. Because this link is trusted, DB2 accepts the distributed identity and tells RACF to search for the RACMAP to match the distributed user. Once verified, RACF passes back user ID SIMOND to DB2. SIMOND inherits the access from the DB2 role and is able to access the table.

ROGERIO is used to establish the connection only, and might not necessarily have access to any DB2 objects. It needs access to the DIST profile (for remote connectivity) if defined. ROGERIO can be known as the *network* ID. Its main function is to enable the trusted connection.

With the RACF RACMAP definition, the distributed identity is mapped to an authorized user ID, in this example, SIMOND, which can be known as a system or application user ID.

Using *trusted context* on a distributed thread tells DB2 to trust this link and allow the distributed user to access DB2. After DB2 receives the incoming request on a trusted connection, it passes control to RACF to verify that the mapping exists in the IDIDMAP class.

EMPREAD, the DB2 role, might not be required. If so, SIMOND needs to have the explicit authority to DB2 objects. In this example, SIMOND does *not* have access to table EMP. But, because of the association to DB2 role, EMPREAD, it is able to access table EMP.

This clearly shows you how a distributed identity, which is unknown to the database server, is able to make a connection. In most systems today, the distributed user is unknown to z/OS, where all threads access DB2 via a common application ID. Although this is still the case within a trusted context, we are now able to associate a distributed user ID and the DB2 for z/OS threads. What is most significant is that the mapping occurs in RACF, and therefore the association is managed within a secure environment.

Note: Refer to Section 10.6.1, “Identity propagation,” of the IBM Redbooks publication *DB2 10 for z/OS Technical Overview*, SG24-7892, which displays the coding of the sample application.

We show the steps taken to map the distributed identity to RACF and to verify the results in an audit report. The following steps were performed to achieve identity propagation in this example:

1. RACFMAP - required for RACF mapping.
2. Create the DB2 trusted context.
3. Create the DB2 role (optional).
4. Enable RACF/DB2 exit (optional).
5. Execute the application, display thread output, and produce audit reports.

Note: At the time of writing, the length of the user ID in DB2 was limited to 8-character bytes. However, this restriction has been lifted by APAR PM31429. Because the distributed identity is not a defined RACF user ID, DB2 will not impose any length restrictions on the user ID (which might contain a distinguished name in the x.500 format).

9.2 RACMAP command

Before issuing the RACMAP command, ensure that you have RACF class IDIDMAP activated. In Figure 9-2, distributed user ‘JoesData’, known only to the distributed world, is mapped to RACF user ID, SIMOND.

```
RACMAP ID(SIMOND) MAP USERDIDFILTER(NAME('JoesData')) REGISTRY(NAME('*'))
```

Figure 9-2 RACMAP command

The RACMAP command can be used to map many distributed users to a single z/OS user ID.

The registry name, in this example, is set up as wildcard to allow any registry to connect. The actual registry name is not verified, but it is used for mapping the distributed identity within RACF. However, the distributed application must send the registry name, wildcard or not, or

else DB2 will not recognize the distributed identity, and pass the user information to RACF as a normal user ID, and in this case would cause an abend.

9.3 Creating DB2 trusted context

In Figure 9-3 we create the trusted context CTXIDID, with ROGERIO being the system authid, and also supply the connection attribute, IP address, that is the client's IP address in this example. The combination of system authid and IP address enables the trusted connection in DB2.

```
CREATE TRUSTED CONTEXT CTXIDID
BASED UPON CONNECTION USING SYSTEM AUTHID ROGERIO
ATTRIBUTES (ADDRESS('9.12.5.148'))
DEFAULT FOLE DEFREAD
WITHOUT ROLE AS OBJECT OWNER
ENABLE
WITH USE FOR SIMOND ROLE EMPREAD ;
```

Figure 9-3 Create trusted context command

SIMOND is the authorization ID to which the distributed identity gets mapped. SIMOND might not be known to the remote system and can be controlled by the DB2 and RACF administrators.

9.4 Creating the DB2 role

In this example, DB2 roles are used to test how it can be associated in a trusted context. We mention that this is optional because SIMOND can have access to required DB2 objects. In our test scenario, SIMOND does not have authority in DB2. Example 9-1 shows the commands issued to create the ROLE and grant access to the ROLE.

Example 9-1 Create role

```
CREATE ROLE EMPREAD;
GRANT SELECT ON TABLE DSN81010.EMP TO ROLE EMPREAD ;
```

9.5 RACF/DB2 exit (optional)

In our example, we show usage with and without the RACF/DB2 exit, RACF/DB2 exit, and DB2 roles.

Roles in RACF

If you are not using the RACF/DB2 exit, RACF would not have any reference to the role ID. DB2 would associate the role in the trusted context, and there is no need for RACF to be involved.

Using RACF/DB2 exit, the authority to DB2 objects is verified in RACF. Therefore, the role must be associated in RACF, and therefore must exist in RACF.

The association of the role to a DB2 object resource profile is done using the RACF PERMIT. Example 9-2 displays the command used.

Example 9-2 DB2 roles association in RACF

```
RDEFINE MDSNTB DBOD.DSN81010.EMP.SELECT UACC(NONE)
PERMIT DBOD.DSN81010.EMP CLASS(MDSNTB) ID(SIMOND) ACCESS(READ)
  WHEN(CRITERIA(SQLROLE(EMPREAD)))
```

In Example 9-2, SIMOND is permitted READ access to the profile only when SIMOND has been associated with role EMPREAD. If SIMOND is not working through a trusted context that has this role associated, SIMOND gets a -551, access denied.

9.6 Executing the sample Java application

The sample Java application performs a connection to DB2 for z/OS, establishes a trusted connection, and returns to the application when prompting the user to press the Enter key before the application continues. When you press Enter to continue, the application switches the user ID to the distributed identity, accesses DB2, and returns the result.

Example 9-3 is an example of the first section of the application.

Example 9-3 Initial connect to DB2 to establish trusted connection

```
String user = new String("ROGERIO");
String password = new String("rogerpsw");
com.ibm.db2.jcc.DB2ConnectionPoolDataSource ds1 =
new com.ibm.db2.jcc.DB2ConnectionPoolDataSource();
ds1.setServerName("wtsc58.itso.ibm.com");
ds1.setPortNumber(38390);
ds1.setDatabaseName("DB0D");
ds1.setDriverType (4);
java.util.Properties properties = new java.util.Properties();
try
{objects = ds1.getDB2TrustedPooledConnection(
user,password, properties);}
catch(Exception ex)
```

In Example 9-3 on page 117, the program performs a connect using `getDB2TrustedPooledConnection`, and DB2 verifies the user and password in RACF, then based on the trusted context definition, returns with a trusted connection. Figure 9-4 is the output from the `DISPLAY THREAD` command. It shows that `ROGERIO` is the authid, using trusted context `CTXIDID`. Further down in the report, notice the IP address, which was used in the connection attribute in the trusted context definition. This confirms that a trusted context has been selected, and therefore this thread is a trusted connection.

```

DSNV401I  -DBOD DISPLAY THREAD REPORT FOLLOWS -
DSNV424I  -DBOD INACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID   PLAN     ASID  TOKEN
SERVER   R2      0 db2jcc_appli  ROGERIO  DISTSERV 0076   235
V485-TRUSTED CONTEXT=CTXIDID,
        SYSTEM AUTHID=ROGERIO,
        ROLE=DEFREAD
V437-WORKSTATION=thumbiran, user ID=rogerio,
        APPLICATION NAME=db2jcc_application
V441-ACCOUNTING=JCC04070thumbiran
V445-G90C0594.GBCO.C7450E61D3FC=227 ACCESSING DATA FOR
( 1)::9.12.5.148
V447--INDEX SESSID          A ST TIME
V448--( 1) 38390:3008      W R2 1103218084741
DISPLAY INACTIVE REPORT COMPLETE
DSN9022I  -DBOD DSNVDT '-DIS THREAD' NORMAL COMPLETION

```

Figure 9-4 `DISPLAY THREAD` - First section

`DEFREAD` is the default DB2 role, set up from the `CREATE TRUSTED` command. Refer to 9.3, “Creating DB2 trusted context” on page 116. `DEFREAD` can have little or no DB2 authorization, depending on rules set up within each organization.

The second part in the application switches the user ID after we press Enter. Figure 9-5 is the code to perform this switch.

In Figure 9-5, which is the second part of the application, we switch the user IDs by populating `user1` with the distributed identity ‘`JoesData`’ and registry with ‘`Registry01`’.

```

String user1 = new String("JoesData");
String user1_pwd = new String(" ");
String registry = new String("Registry01");
String countr = new String(" ");
try{
con = pooledCon.getDB2Connection(
cookie,user1, user1_pwd, registry,
userSecTkn, originalUser, properties);
stmt = con.createStatement();
ResultSet rs1 = stmt.execute("select count(*) from dsn81010.emp;");
// retrieve and display

```

Figure 9-5 Switch of user ID in a trusted connection

By using the `getDB2Connect` method we attempt the logon to DB2 for z/OS. DB2 would pass on the distributed identity to RACF to get mapped. Once mapped, the user ID is switched in DB2 to the mapped user ID, `SIMOND`.

Figure 9-6 shows a display thread after we execute the second part of the application.

```
DSNV401I -DBOD DISPLAY THREAD REPORT FOLLOWS -
DSNV424I -DBOD INACTIVE THREADS -
NAME ST A REQ ID AUTHID PLAN ASID TOKEN
SERVER R2 0 db2jcc_appli SIMOND DISTSERV 0076 235
V485-TRUSTED CONTEXT=CTXIDID,
      SYSTEM AUTHID=ROGERIO,
      ROLE=EMPREAD
V437-WORKSTATION=thumbiran, user ID=rogerio,
      APPLICATION NAME=db2jcc_application
V441-ACCOUNTING=JCC04070thumbiran
V445-G90C0594.GBC2.C74511EBA293=235 ACCESSING DATA FOR
( 1)::9.12.5.148
V447--INDEX SESSID A ST TIME
V448--( 1) 38390:3010 W R2 1103218260975
DISPLAY INACTIVE REPORT COMPLETE
DSN9022I -DBOD DSNVDT '-DIS THREAD' NORMAL COMPLETION
```

Figure 9-6 Display thread displays switch user ID

In Figure 9-6 the AUTHID is now changed from ROGERIO to SIMOND, which was mapped in RACF. Notice that the trusted context is still used, with system authid ROGERIO, trusted context CTXIDID, and the role has changed to EMPREAD.

The application completes successfully, and this clearly displays the user ID having changed to SIMOND. We would reaffirm this later in the audit report.

Audit trace

The audit trace is done using RACF and DB2 for z/OS SMF data. This audit trail will show the distributed identity, through DB2 for z/OS, being recorded under RACF SMF, and also DB2 SMF data. In our example, we use IBM Security zSecure and IBM Omegamon for DB2.

9.7 RACF audit trace

In this section we list zSecure reports on RACF SMF Type 80 records using Consul Auditing and Reporting Language (CARLa) scripts to show an audit trail.

Figure 9-7 is an JCL example for a job that executes a CARLa script (contained in the SYSIN DD statement). Filters are set for class, event, and jobname.

```
//REPORT EXEC PGM=CKRCARLA
//STEPLIB DD DISP=SHR,DSN=CKR.SCKRLOAD
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  suppress msg=1400
  suppress racf ioconfig

  alloc smf

  newlist type=smf
  select event=allcmds class=(dsnr,dsnadm,mdsntb)
  select event=(racmap,setropts)
  select class=(dsnr,dsnadm,mdsntb) event=access
  select jobname=db0d*
  sortlist date(9) time(8) type user ID,
           event eventqual class intent resource,
           / "(22) auth_user_name, /* 424 */
           / "(22) auth_user_regname /* 425 */
/*
```

Figure 9-7 zSecure CARLa script

Figure 9-8 shows a report generated from the zSecure script.

Typ	User	Event	Eq	Class	Intent	Resource
80	ROGERIO	ACCESS	0	DSNR	READ	DB0D.DIST
80	SIMOND JoesData	RACINIT	12	USER		SIMOND
80	SIMOND JoesData	ACCESS	0	DSNR	READ	DB0D.DIST
80	SIMOND JoesData	ACCESS	0	MDSNTB	READ	DB0D.DSN81010.EMP

Figure 9-8 zSecure report

There are four records displayed in this report to indicate that distributed identity was used. The first record shows ROGERIO accessing DB2 through RACF Class DSNR and read access to profile DB0D.DIST. At this point, the trusted connection has been established.

The second record is a RACINIT. The user ID SIMOND, which is the new AUTHID, and the distributed identity JoesData are displayed, with the distributed identity being associated with SIMOND.

The third record shows SIMOND having read access to the DB0D.DIST profile. Again, you can see JoesData attached to SIMOND.

The fourth record shows SIMOND getting access to the table resource profile in RACF. This would only be displayed when the RACF/DB2 exit is used, and also if the dynamic statement cache does not have this statement cached. Once again, you see the distributed identity, JoesData, attached to SIMOND.

Remember that with the RACF/DB2 exit, the SIMOND is also associated with SQLROLE EMPREAD, which allows him to get access to the table profile (Example 9-2 on page 117).

9.8 DB2 Audit trail

Using Omegamon for DB2 and AUDIT POLICY, we display the events that occur in DB2, relating to the distributed identity. Figure 9-9 shows the AUDIT policy created for this example.

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME,OBJECTSCHEMA,OBJECTNAME,OBJECTTYPE,VALIDATE)
VALUES('VALIDTE','DSN81010','EMP','T','A') ;

-START TRACE(AUDIT) AUDTPLCY(VALIDTE) DEST(SMF)
```

Figure 9-9 DB2 Audit policy

Figure 9-10 shows the output from the Omegamon report.

```
ROGERIO db2jcc_a DRDA 16:38:54.35 AUTHCHG TYPE: ESTABLISH TRUSTED CONTEXT
ROGERIO ppli C7473AF762CE OBJECT OWNER: AUTHID
DISTSERV SERVER SECURITY LABEL:
REQLOC :::9.12.5.148 CONTEXT NAME: CTXIDID
ENDUSER :rogerio CONTEXT ROLE: DEFREAD
WSNAME :thumbiran USER ROLE:
TRANSACTION:db2jcc_application PREV. SYSAUTHID: ROGERIO
REUSE AUTHID:
SERVAUTH NAME:
JOB NAME:
ENCRYPTION: NONE
TCP/IP USED:9.12.5.148
```

Figure 9-10 Establish trusted context

In Figure 9-10, we executed the application to establish the trusted connection. The context name is CTXIDID and Sysauthid is ROGERIO.

In Figure 9-11, the audit trail shows the distributed identity as derived local UID, JoesData. This is the first account of the distributed identity, which was propagated over the thread initiated by user ROGERIO.

```

ROGERIO db2jcc_a DRDA          16:38:56.58 AUTHCHG TYPE:          N/P
ROGERIO ppli      C7473AF762CE          IP ADDR:          90C0594
DISTSERV SERVER                                DERIVED LOCAL UID: JoesData
REQLOC  :::9.12.5.148
ENDUSER :rogerio
WSNAME  :thumbiran
TRANSACT:db2jcc_application

```

Figure 9-11 Distributed identity

In Figure 9-12, DB2 issues an end of identity event, because new sqlid SIMOND has been assigned to this thread. Notice that distributed identity JoesData is associated with SIMOND.

```

SIMOND db2jcc_a DRDA          16:38:56.58 AUTHCHG TYPE:          END OF IDENTIFY
JoesData ppli      C7473AF762CE          PREVIOUS AUTHID: JoesData
DISTSERV SERVER                                SECONDARY AUTHID: SYS1
REQLOC  :::9.12.5.148                                STATUS: SUCCESS
ENDUSER :rogerio                                CURRENT SQLID: SIMOND
WSNAME  :thumbiran
TRANSACT:db2jcc_application

```

Figure 9-12 End of identity event

In Figure 9-13, DB2 reuses the thread after the application executes the SQL statement. The user role is EMPREAD and the reuse authid is SIMOND.

```

SIMOND db2jcc_a DRDA          16:38:56.58 AUTHCHG TYPE: REUSE TRUSTED CONTEXT
IBMUSER ppli      C7473AF762CE          OBJECT OWNER: AUTHID
DISTSERV SERVER                                SECURITY LABEL:
REQLOC  :::9.12.5.148                                CONTEXT NAME: CTXIDID
ENDUSER :rogerio                                CONTEXT ROLE: DEFREAD
WSNAME  :thumbiran                                USER ROLE: EMPREAD
TRANSACT:db2jcc_application                    PREV. SYSAUTHID:ROGERIO
                                                REUSE AUTHID: SIMOND
                                                SERVAUTH NAME:
                                                JOB NAME:
                                                ENCRYPTION:
                                                TCP/IP USED:

```

Figure 9-13 Reuse trusted context



Identity propagation using CICS Web services

This chapter describes a scenario for identity propagation using CICS Web services and WebSphere DataPower.

10.1 Scenario overview

This chapter describes a scenario involving a web service call between two companies. The service provider application is a CICS core banking application. The service requester is an application running on a system of one of the bank's business partners. The trusted partner is authorized to access services such as the posting inquiry service, which returns the most recent account transactions for a banking customer, and the account transfer service, which transfers money from one account to another.

An employee of the business partner makes a web service call that is authenticated by WebSphere DataPower. After successful authentication, WebSphere DataPower propagates the user's distributed identity (in the form of a DN) to the CICS core banking application. The bank has a requirement to authorize requests based on a generic RACF user ID that represents the business partner, but also to keep an audit trail of which business partner employee invoked the service.

Figure 10-1 shows an overview of the CICS Web services identity propagation scenario.

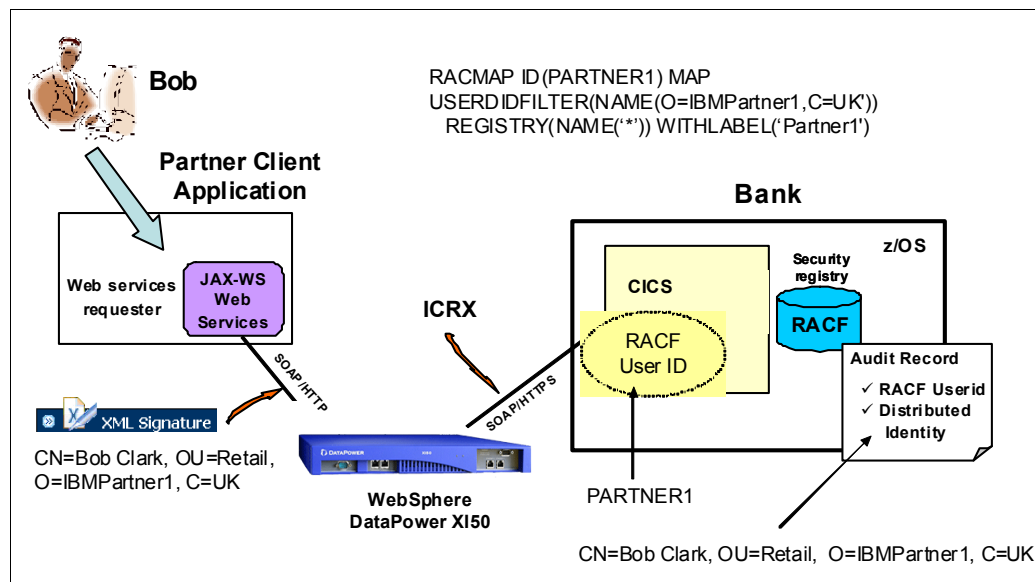


Figure 10-1 CICS Web services identity propagation scenario

The sequence of steps is:

1. A partner employee Bob uses a partner client application that sends a web service request to the bank.
2. The service requester application generates a BinarySecurityToken element from Bob's X.509 certificate, signs the message with Bob's private key, and sends the request to the target endpoint, which is configured to be the DataPower appliance.
3. DataPower verifies the XML digital signature.

Note: XML-Signature Syntax and Processing (XML digital signature) is a specification that defines XML syntax and processing rules to sign and verify digital signatures for digital content. The specification was developed jointly by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).

4. DataPower extracts the identity of the service requester (CN=Bob Clark, OU=Retail, O=IBMPartner1, C=UK) using the certificate passed as part of the <X509/> element of the digitally signed message.
5. DataPower authenticates the user by validating the signer certificate of the digitally signed message.
6. DataPower propagates Bob's identity (Bob's DN) to CICS in the form of an ICRX over a trusted SSL connection.

Note: We do not provide information about configuring SSL in this chapter. For detailed information about configuring SSL with CICS, refer to the IBM Redbooks publication *Securing CICS Web Services*, SG24-7658.

7. CICS receives the SOAP message from DataPower. The PIPELINE configuration file includes the CICS-supplied WS-Security handler program, which locates the ICRX in the WS-Security header and uses the ICRX to identify the user.
8. CICS issues a RACROUTE REQUEST=VERIFY to map the ICRX into the RACF user ID PARTNER1.
9. The CICS task runs under the mapped RACF user ID (PARTNER1) but retains the association with the original distributed identity (CN=Bob Clark, OU=Retail, O=IBMPartner1, C=UK).

10.2 Architectural overview

Figure 10-2 shows the topology used to test this scenario.

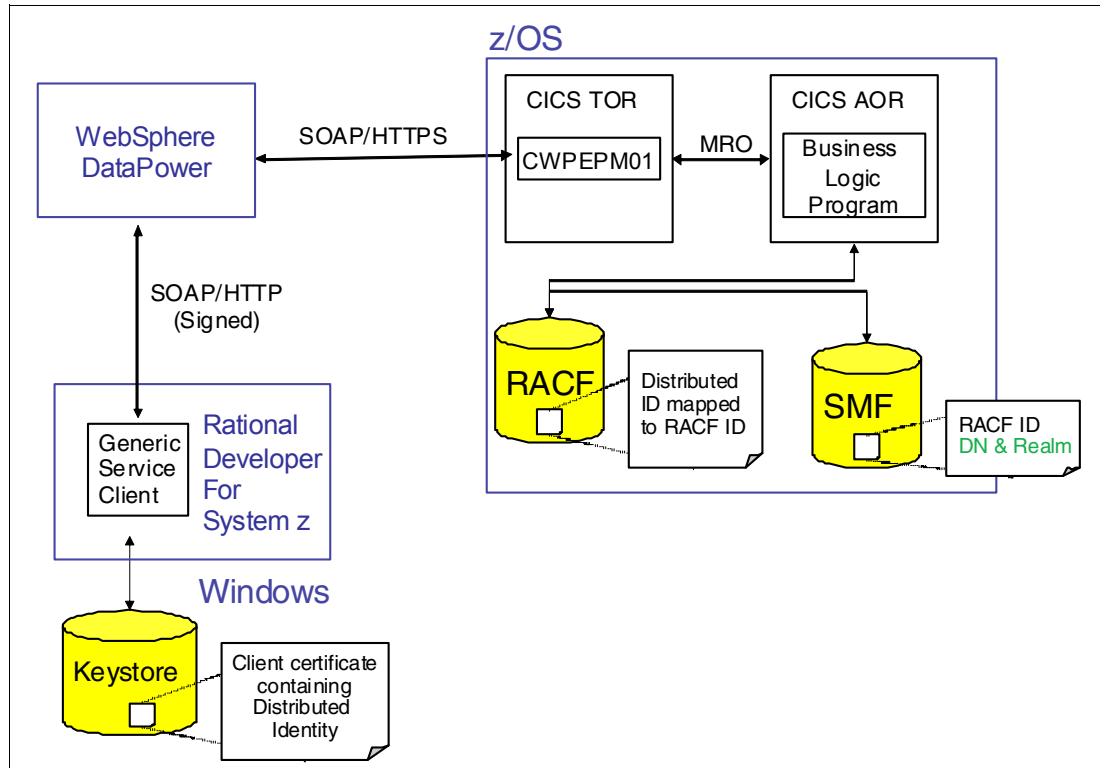


Figure 10-2 CICS Web services topology used in this scenario

The general flow of events is:

1. The Generic Service Client of Rational Developer for System z (RDz) is used to simulate the partner service requester that makes a web service call to the account transfer service. RDz is configured to access a keystore that contains the X.509 certificate, which is used to sign the SOAP message that is sent to WebSphere DataPower.
2. After authenticating the service requester and extracting the user's identity, DataPower forwards the request to CICS with the distributed identity contained in an ICRX that is transported as part of the WS-Security header in the SOAP message.
3. The CICS Terminal Owning Region (TOR) processes the SOAP message, extracts the ICRX token from the WS-Security header, and calls RACF to map the distributed identity to a RACF user ID.
4. The CICS Web service wrapper program (CWPEPM01) runs in the TOR under the mapped RACF user ID.
5. Program CWPEPM01 links to the business logic program that runs in the CICS Application Owning Region (AOR). Business logic programs in the AOR also run under the mapped RACF user ID.
6. The original distributed identity (DN and realm) can be audited in both the TOR and AOR, and is also available as part of the CICS task association data when using the CICS Explorer™ to monitor running tasks.

10.3 Preparation

This section contains reference information about product versions and configuration values used in the CICS Web services scenario.

10.3.1 Software versions

Table 10-1 lists the software versions used in this scenario.

Table 10-1 Software and firmware used in the CICS Web services scenario

Windows	DataPower	z/OS
Rational Developer for System z V7.6.2.1	Firmware version XI50.3.8.0.3 / Build:182012	z/OS V1.12 with RACF APAR OA34258 and SAF APAR OA34259
Windows Server 2003 Enterprise Edition SP2		CICS Transaction Server V4.1with identity propagation enabling APARs PK83741, PK95579, PM01622, and PK98426.

10.3.2 IP addresses and ports

Table 10-2 lists the IP addresses and ports used in this scenario.

Table 10-2 IP addresses and ports used in the CICS Web services scenario

Value	CICS TS	DataPower
IP address	9.212.128.20	9.212.130.128
TCP/IP port	61001	9080

10.3.3 CICS resource definition checklist

Table 10-3 lists the CICS TOR definitions used in this scenario.

Table 10-3 CICS TOR resource definitions used in the CICS Web services scenarios

Resource	Value
APPLID	CICSRT10
TCPIPSERVICE	SSLWEBS
PIPELINE	PIPFIS
WEBSERVICE	TranExt
URIMAP	UR9ZTE
Pipeline configuration file	SOAP12provider.xml

10.3.4 User IDs

Table 10-4 lists the user IDs that we used in our configuration.

Table 10-4 User IDs used in DataPower scenario

Distributed identity	RACF user ID
CN=Bob Clark, OU=Retail, O=IBMPartner1, C=UK	PARTNER1
CN=Nigel Williams, OU=Commercial, O=IBMPartner1, C=UK	PARTNER1
CN=Alain Roessle, OU=Banque Internationale, O=IBMPartner2, C=FR	PARTNER2
CN=Fabrice Jarassat, OU=Banque Internationale, O=IBMPartner2, C=FR	PARTNER2
CN=Alice Smith, OU=New York Branch, O=IBMPartner3, C=US	PARTNER3
CN=Jessica Jones, OU=Miami Branch, O=IBMPartner3, C=US	PARTNER3

10.3.5 Keystore and certificates

Each distributed user ID has an X.509 certificate that is stored in a Java keystore called SmarterBankingShowcaseKeystore.jks. Example 10-1 shows a listing of the certificate that is used for signing the SOAP message in our test scenario.

Example 10-1 Listing of signer certificate

```
Alias name: cn=bob clark,ou=retail,o=ibmpartner1,c=uk
Creation date: Apr 6, 2011
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=Bob Clark, OU=Retail, O=IBMPartner1, C=UK
Issuer: CN=SmarterBankingShowcaseCA, OU=pssc, O=ibm, C=FR
Serial number: 4d9c204200028919
Valid from: 4/6/11 10:11 AM until: 4/5/12 10:11 AM
Certificate fingerprints:
    MD5: 01:29:CC:63:50:A9:50:91:24:3A:09:51:AD:17:EA:9F
    SHA1: B7:26:31:44:6D:54:D8:CC:3F:C4:C5:B0:9E:08:36:4B:C0:CE:71:7C
Certificate[2]:
Owner: CN=SmarterBankingShowcaseCA, OU=pssc, O=ibm, C=FR
Issuer: CN=SmarterBankingShowcaseCA, OU=pssc, O=ibm, C=FR
Serial number: 4d9c1f1300022e6f
Valid from: 4/6/11 10:06 AM until: 4/5/14 10:06 AM
Certificate fingerprints:
    MD5: 04:7A:23:98:4F:AA:5F:B0:19:A2:AA:4B:A7:CD:6E:96
    SHA1: 1D:01:1A:AF:A7:A1:93:2C:55:3E:7B:6C:10:BD:24:09:16:13:98:20
```

Note the following information about the bolded information in Example 10-1:

- ▶ The alias name for the signer certificate is **cn=bob clark,ou=retail,o=ibmpartner1,c=uk**.
- ▶ The owner of the signer certificate is **CN=Bob Clark, OU=Retail, O=IBMPartner1, C=UK**.
- ▶ The issuer of the signer certificate is **CN=SmarterBankingShowcaseCA, OU=pssc, O=ibm, C=FR**.

Example 10-2 shows a listing of the issuer certificate.

Example 10-2 Listing of issuer certificate

Alias name: **cn=showcaseca,ou=pssc,o=ibm,l=montpellier,c=fr**
Creation date: Apr 7, 2011
Entry type: trustedCertEntry

Owner: **CN=showcaseCA, OU=PSSC, O=IBM, L=Montpellier, C=FR**
Issuer: CN=showcaseCA, OU=PSSC, O=IBM, L=Montpellier, C=FR
Serial number: 4c400d92000b1b25
Valid from: 7/16/10 9:43 AM until: 7/16/11 9:43 AM
Certificate fingerprints:
 MD5: 42:3F:6D:8F:38:FD:7D:40:57:7D:05:46:7A:6C:59:77
 SHA1: D8:AA:6D:2B:D6:75:C8:98:01:5B:CC:58:7F:A8:B6:B4:7C:7B:CB:73

Note the following information about the bolded information in Example 10-2:

- ▶ The alias name for the issuer certificate is **cn=showcaseca,ou=pssc,o=ibm,l=montpellier,c=fr**.
- ▶ The owner of the issuer certificate is **CN=showcaseCA, OU=PSSC, O=IBM, L=Montpellier, C=FR**.
- ▶ The issuer certificate is self-signed.

10.4 Configuring RDz

In this section we show you how to configure the Generic Service Client of Rational Developer for System z (RDz) to send a signed SOAP message.

The Generic Service Client is a component that enables users to interact with web services under test. Users point the Generic Service Client to the service interface definition, such as a Web Services Description Language (WSDL) file. The Generic Service Client then automatically and dynamically creates a graphical user interface that exposes the operations of that service.

Figure 10-3 shows an invocation of the account transfer Web service WSDL (CWPEPM01.wsdl) using the Generic Service Client.

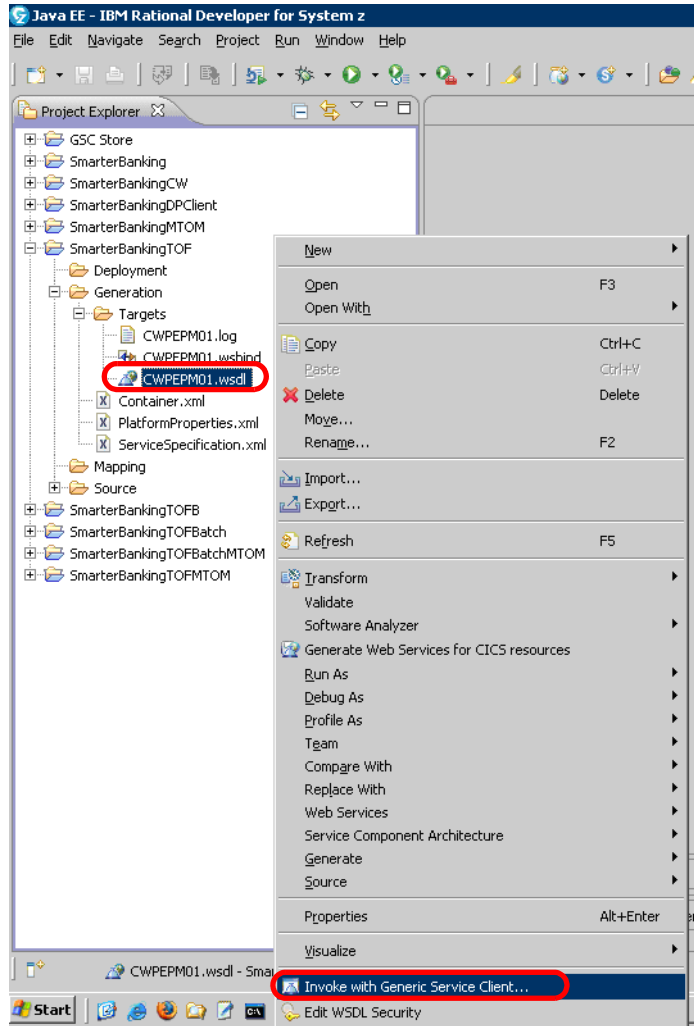


Figure 10-3 Invoke account transfer service using Generic Service Client

After invoking the Generic Service Client we are able to fill in values for the different elements of the SOAP request message (Figure 10-4). The account transfer service, for example, requires input for elements including posting date, account numbers, and the amount of money to transfer.

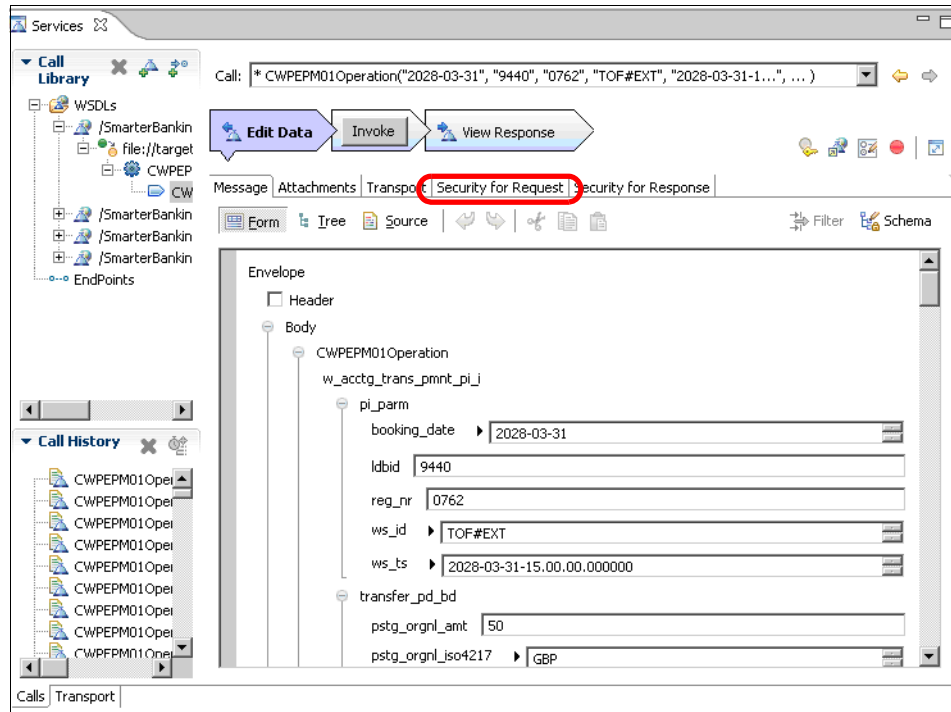


Figure 10-4 Entering data for the account transfer service

The Security for Request tab is then used to edit the SOAP message security algorithms to be applied to the service request before it is sent. Figure 10-5 shows the XML signature that is configured for the account transfer service request.

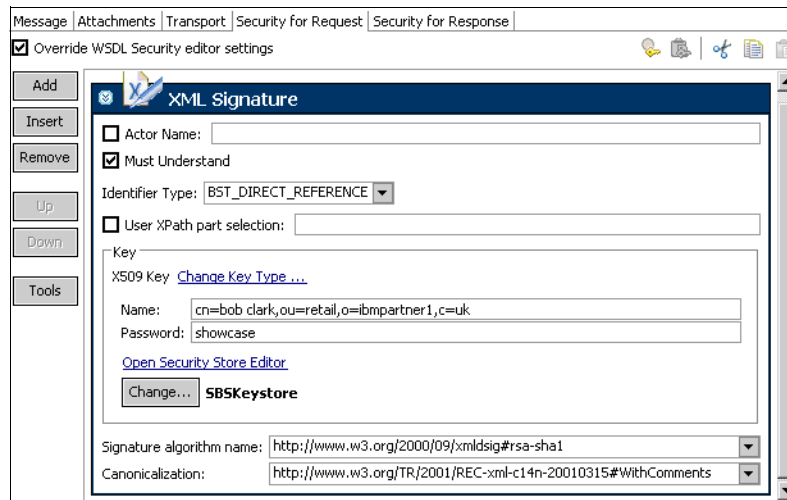


Figure 10-5 XML signature for the account transfer service

The XML signature (Figure 10-5 on page 131) was configured as follows:

1. We select **BST_DIRECT_REFERENCE** (Direct reference) as the identifier type because we want to send the public key as part of the complete certificate, because we will use the certificate for identification in DataPower in addition to decrypting the signed message.
2. We chose an X509 Key type and specify the following values for the key:
 - **cn=bob clark,ou=retail,o=ibmpartner1,c=uk** as the name (alias) for the key
 - **showcase** as the key password
3. We accept the default signature algorithm and canonical names.

For further information about testing the account transfer service see 10.8, “Testing the scenario” on page 155.

10.5 Configuring WebSphere DataPower

In this section we show you how to configure DataPower, in particular:

- ▶ How to define the web service proxy for the account transfer web service
- ▶ How to verify the signed SOAP message
- ▶ How to configure an authentication, authorization, and auditing (AAA) policy for identity propagation

We document the following procedures:

- ▶ Accessing the DataPower control panel
- ▶ Creating DataPower objects
- ▶ Configuring an HTTP front-side handler
- ▶ Adding certificates to DataPower
- ▶ Configuring a web service proxy
- ▶ Verifying the XML digital signature
- ▶ Configuring the AAA policy for identity propagation

Accessing the DataPower control panel

We use the following URL to access the DataPower control panel:

<https://9.212.130.128:9090/>

Figure 10-6 shows the control panel.

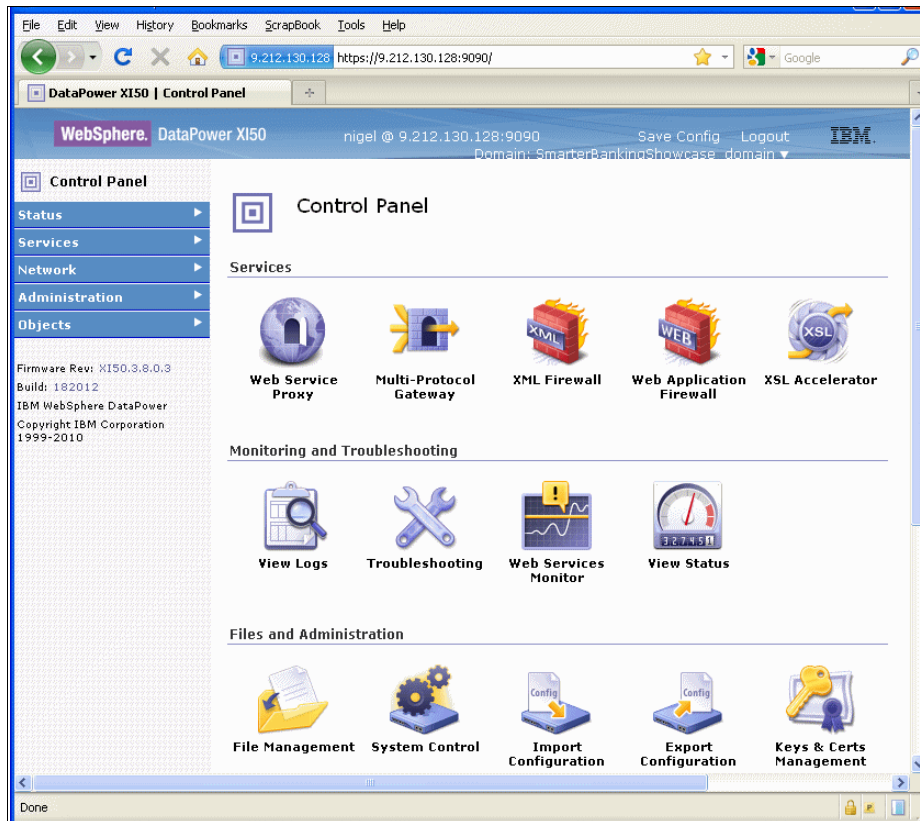


Figure 10-6 DataPower Control panel

DataPower objects

DataPower configuration items are internally represented as objects for reuse. Figure 10-7 outlines the objects that we configured for our scenario.

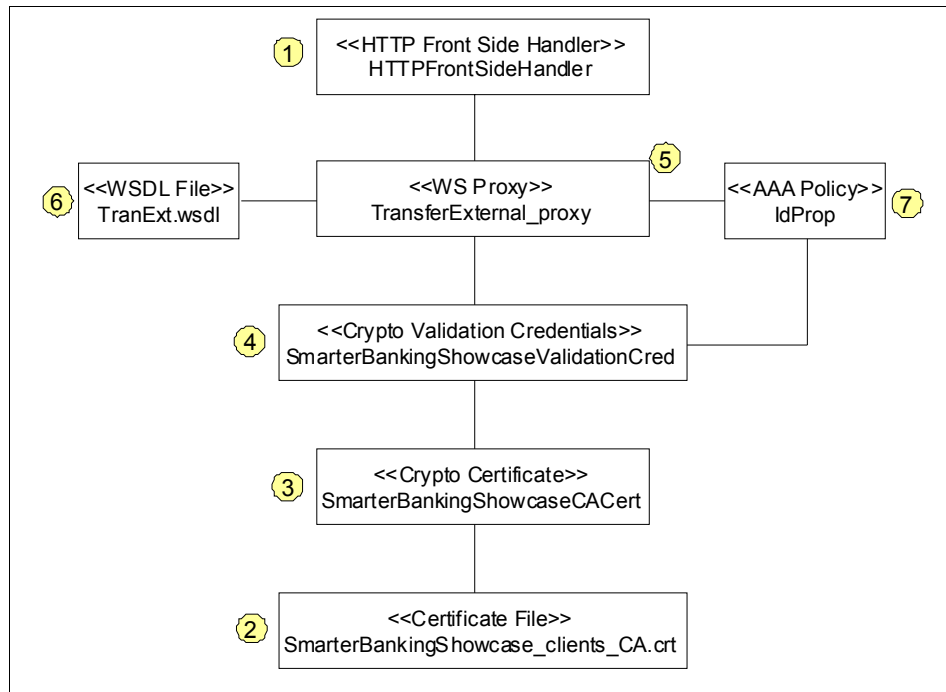


Figure 10-7 DataPower objects

The steps in Figure 10-7 are explained here:

- ▶ An HTTP Front Side Handler (1) object handles HTTP protocol communications with HTTP clients. We create one to handle all HTTP requests. It listens on port 9080.
- ▶ We use the SmarterBankingShowcase_clients_CA.crt (2) file that contains the root certificate associated with the signer certificate, and we create a Crypto Certificate (3).
- ▶ A Crypto Validation Credential (4) includes a collection of certificate objects used for authenticating presented credentials and verifying XML digital signatures.
- ▶ We create a WS Proxy (5) using the TranExt.wsdl (6) that intercepts web service calls to the CICS account transfer service.
- ▶ We associate a verify signature action (7) with the WS Proxy.
- ▶ We associate an AAA policy (8) with the WS Proxy.

Configuring an HTTP front side handler

An HTTP front side handler object handles HTTP protocol communications. We add a local endpoint handler using the following steps:

1. On the DataPower control panel, expand **Objects**, and then click **HTTP Front Side Handler** under the Protocol Handlers section.

The list of HTTP front side handlers is displayed. It should be empty at this stage (Figure 10-8).



Figure 10-8 DataPower - Configure HTTP Front Side Handler

2. Click **Add**.
3. On the HTTP Front Side Handler page, specify the properties of the local endpoint handler:
 - a. Enter HTTPFrontSideHandler in the Name field.
 - b. Enter port number 9080 in the Port Number field.
 - c. Leave the other properties as the default.

Figure 10-9 shows our configured HTTP Front Side Handler.

Name	HTTPFrontSideHandler *
Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	
Local IP Address	0.0.0.0 <input type="button" value="Select Alias"/> *
Port Number	9080 *
HTTP Version to Client	HTTP 1.1
Allowed Methods and Versions	<input checked="" type="checkbox"/> HTTP 1.0 <input checked="" type="checkbox"/> HTTP 1.1 <input checked="" type="checkbox"/> POST method <input type="checkbox"/> GET method <input checked="" type="checkbox"/> PUT method <input type="checkbox"/> HEAD method <input type="checkbox"/> OPTIONS <input type="checkbox"/> TRACE method <input type="checkbox"/> DELETE method <input checked="" type="checkbox"/> URL with Query Strings <input checked="" type="checkbox"/> URL with Fragment Identifiers <input type="checkbox"/> URL with .. <input type="checkbox"/> URL with cmd.exe
Persistent Connections	<input checked="" type="radio"/> on <input type="radio"/> off
Compression	<input type="radio"/> on <input checked="" type="radio"/> off
Maximum Allowed URL Length	16384
Maximum Allowed Total Header Length	128000
Maximum Number of HTTP Request Headers Allowed	0
Maximum Allowed Length of HTTP Header Name	0
Maximum Allowed Length of HTTP Header Value	0
Maximum Allowed Length of HTTP Query String	0
Access Control List	(none) <input type="button" value="+"/> <input type="button" value="..."/>

Figure 10-9 DataPower HTTP Front Side Handler

Note: In Figure 10-9 we show an HTTP front side handler. When configuring an SSL connection from DataPower to CICS, create an HTTPS (SSL) front side handler.

4. Click **Apply**.

Adding certificates to DataPower

Next we create a Crypto Certificate by importing the issuer certificate to DataPower, and we create the Crypto Validation Credentials.

Configuring a Crypto Certificate

A Crypto Certificate defines a certificate object that specifies a public key and key alias. When DataPower receives a signed SOAP request, it validates the signature and also the validity of the certificate, particularly the trust associated with this certificate.

Using the following steps, we load the issuer certificate in DataPower:

1. On the DataPower control panel, expand **Objects**, and then click **Crypto Certificate** under the Crypto section.

The list of Crypto Certificates is displayed. It should be empty at this stage (Figure 10-10).

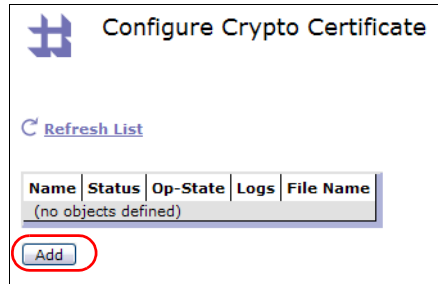


Figure 10-10 DataPower Crypto Certificates

2. Click **Add**.
3. On the Configure Crypto Certificate page:
 - a. Enter SmarterBankingShowcaseCACert in the Name field.
 - b. Use the **Upload** tab to navigate to the location of the issuer certificate and upload the file.
 - c. Enter the password to open the file.
 - d. Click **Apply** to configure the certificate.

Figure 10-10 shows our configured Crypto Certificate.

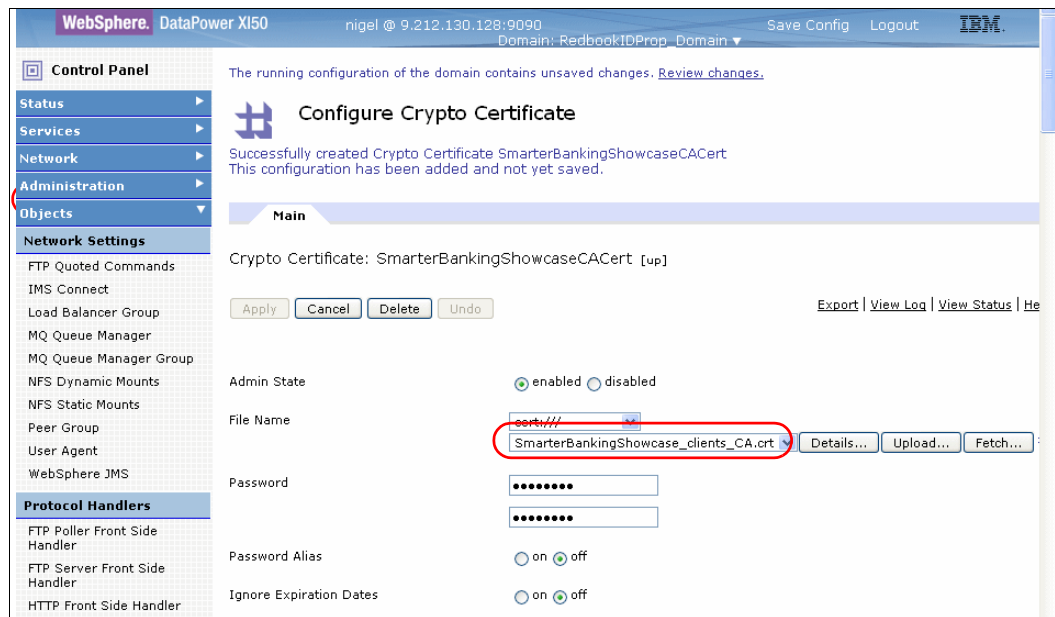


Figure 10-11 DataPower Crypto Certificate

Configuring the Crypto Validation Credentials

A Crypto Validation Credential defines a validation credentials list object that includes a collection of certificate objects used for authenticating presented credentials and validating XML digital signatures.

We create the Crypto Validation Credentials using the following steps:

1. On the DataPower control panel, expand **Objects**, and then click **Crypto Validation Credentials** under the Crypto section.

The list of Crypto Validation Credentials is displayed. It should be empty at this stage (Figure 10-12).

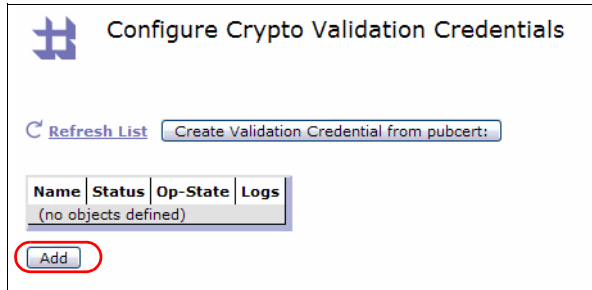


Figure 10-12 DataPower Crypto Validation Credentials

2. Click **Add**.
3. On the Configure Crypto Validation Credentials page, specify the properties of the credentials:
 - a. Enter SmarterBankingShowcaseValidationCred in the Name field.
 - b. Select **SmarterBankingShowcaseCACert** in the Certificates drop-down menu and click **Add** to add the certificate to the credentials list.
 - c. Leave the other properties as the default.
 - d. Click **Apply** to configure the new Crypto Validation Credentials.

Figure 10-13 shows our configured Crypto Validation Credentials.

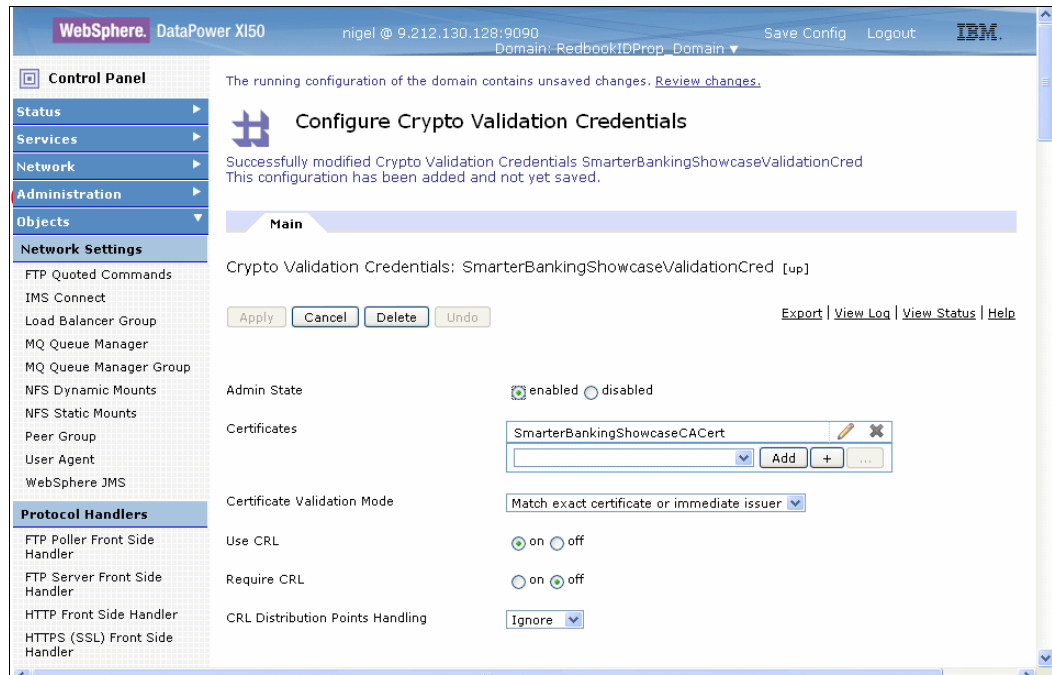


Figure 10-13 DataPower Crypto Validation Credentials

Configuring a web service proxy

In this section, we create a simple web service proxy service that intercepts web service calls to the CICS account transfer service. A web service proxy provides:

- ▶ A facade for arbitrary back-end services.
- ▶ Quick service virtualization by simply uploading WSDL into the DataPower device®.
- ▶ A “living” virtual service that passes messages between the client and the real service so that the client connects to the proxy and not to the back-end service.

In doing this, the service consumer and the service provider do not need to be tightly coupled and bound to each other. The DataPower appliance can hide the details of accessing the service from the consumer.

We create the web service proxy using the following steps:

1. On the DataPower control panel, expand **Services**, and then click **New Web Service Proxy** under the Web Service Proxy section.
2. On the Configure Web Service Proxy panel, enter TransferExternal_proxy for the name of the Web Service Proxy (Figure 10-14).



Configure Web Service Proxy

Web Service Proxy Name

TransferExternal_proxy *

Create Web Service Proxy

Figure 10-14 DataPower Web Service Proxy 1

3. Click **Create Web Service Proxy**.

- To create the WS proxy, we need to upload the WSDL file associated with this WS proxy. On the Configure Web Service Proxy page use the **Upload** tab to navigate to the location of the WSDL file and upload the file (Figure 10-15). Click **Next**.

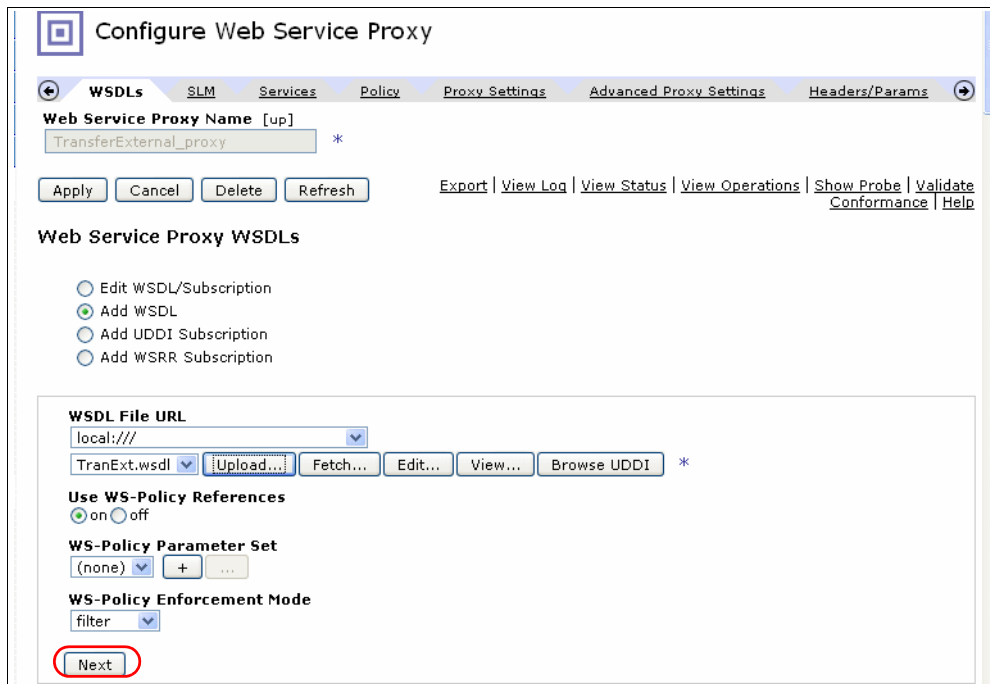


Figure 10-15 DataPower Web Service Proxy 2

- On the next panel (Figure 10-16) we associate the web service proxy with the HTTP front side handler that we configured in “Configuring an HTTP front side handler” on page 135 and we specify the endpoint information for the target CICS account transfer service.

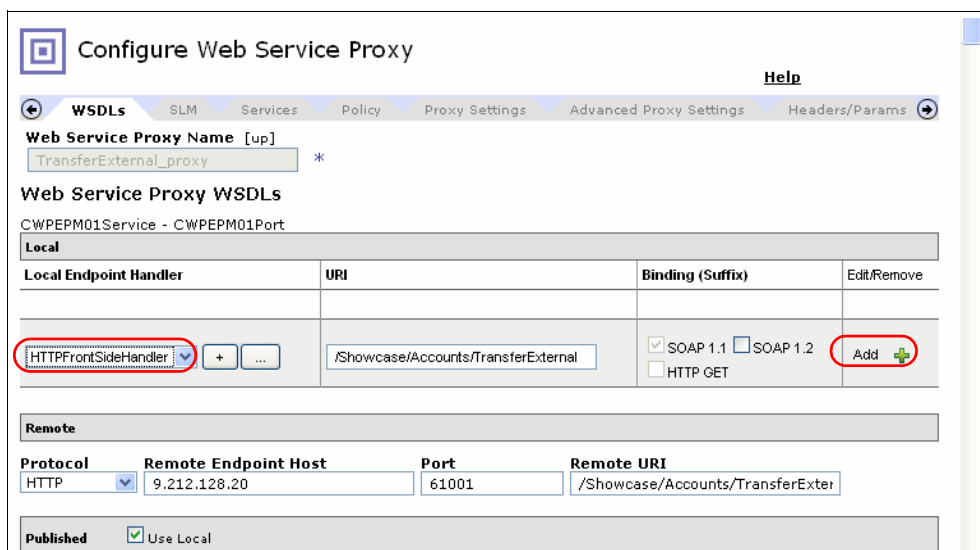


Figure 10-16 DataPower Web Service Proxy 3

Select **HTTPFrontSideHandler** and click the **+Add** link so that it appears in the list of local endpoint handlers. Click **Next** and then **Apply**. Figure 10-17 shows our configured web service proxy.

The screenshot shows the 'Configure Web Service Proxy' window with the following configuration details:

- Web Service Proxy Name:** TransferExternal_proxy
- Web Service Proxy WSDLs:** Edit WSDL/Subscription (selected), Add WSDL, Add UDDI Subscription, Add WSRR Subscription.
- WSDL Source Location:** local:///TranExtWsd
- Endpoint Handler Summary:** 1 up / 1 configured
- WSDL Status:** Okay
- WS-I BP Status:** Okay
- Action:** Remove
- Local Endpoint Handler:** HTTPFrontSideHandler
- URI:** /Showcase/Accounts/TransferExternal
- Binding (Suffix):** SOAP 1.1 (selected), SOAP 1.2, HTTP GET
- Remote Endpoint Host:** 9.212.128.20
- Port:** 61001
- Remote URI:** /Showcase/Accounts/TransferExter

Figure 10-17 DataPower Web Service Proxy 4

Verifying the XML digital signature

Next we configure the signature verification:

1. On the Configure Web Service Proxy page, we click the **Policy** tab. We then expand the port-operation **CWPEPM01Operation** and click the **+ Add Rule** (Figure 10-18).

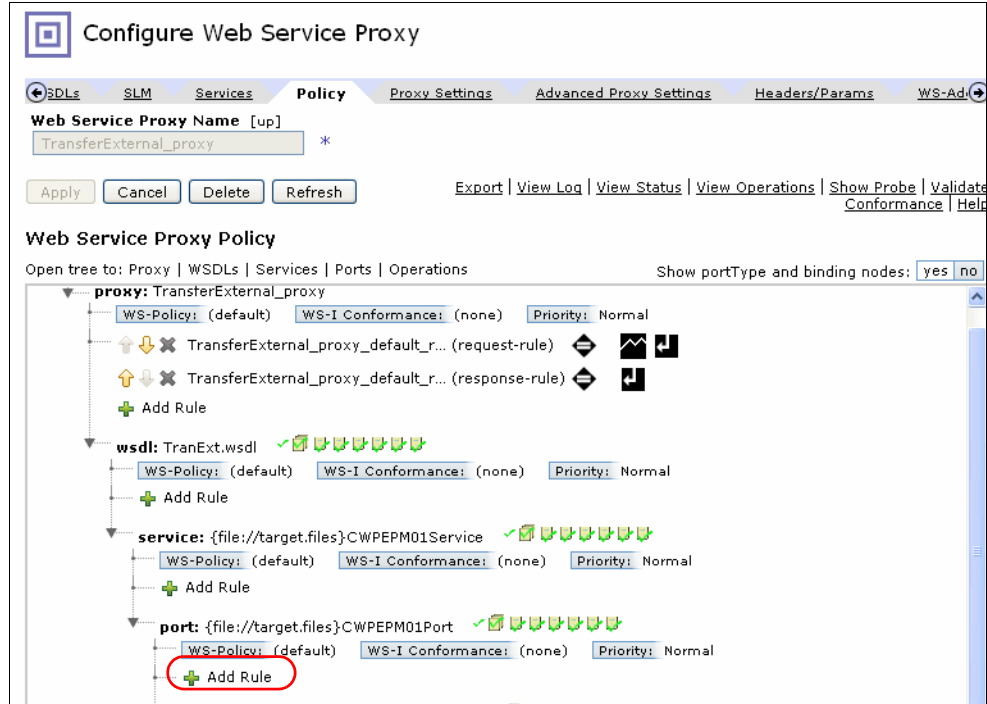


Figure 10-18 DataPower Add Policy Rule

2. The Rule Editor page opens and we select the rule direction, in our case **Client to Server**. We drag and drop the **verify** icon to the right of the equals sign (=) (Figure 10-19).

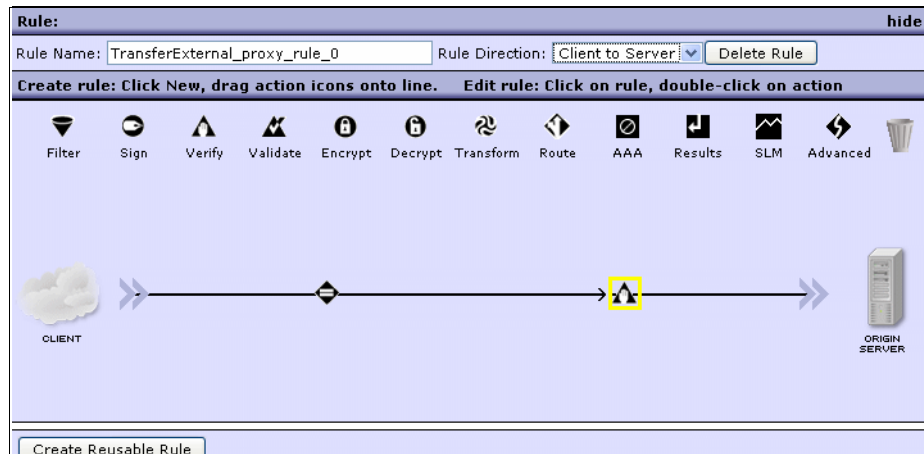


Figure 10-19 DataPower Verify Signature

Note: In the rule editor, when the rule is not configured, there is a bold yellow border around the icon.

3. Double-click the Verify icon. On the Configure Verify Action page, select **INPUT** and chose the **SmarterBankingShowcaseValidationCred** validation credential from the drop-down menu. Click **Done**. Figure 10-20 shows the configured verify action.

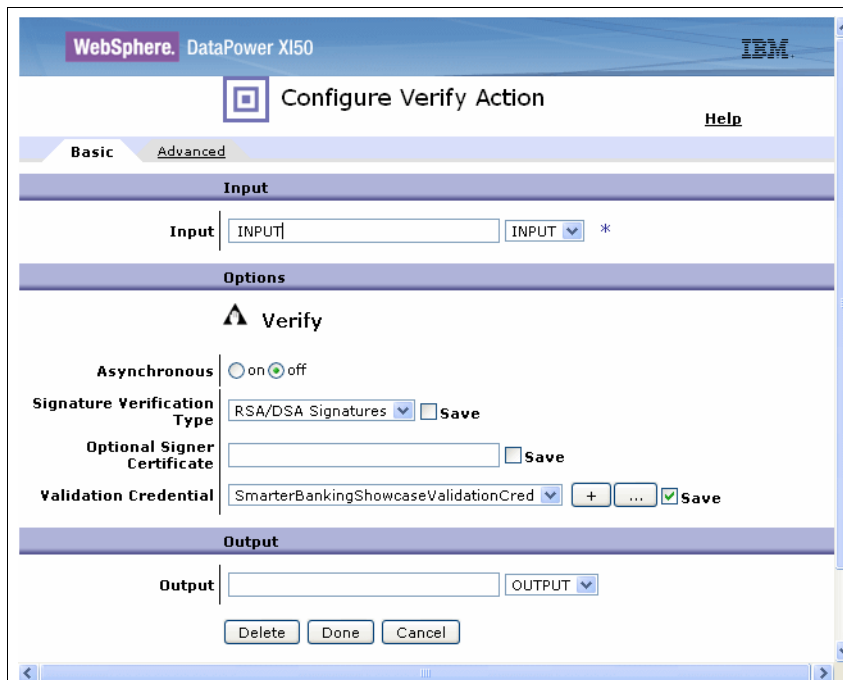


Figure 10-20 DataPower Verify Signature 2

4. Click **Apply** (on the top left of the panel) to apply the configuration.

Configuring AAA policy

In this section we create an AAA policy to secure the account transfer web service. A DataPower AAA policy identifies a set of resources and procedures that can be used to determine whether a requesting client is granted access to a specific service. AAA policies can be considered a type of filter in that they accept or deny a specific client request.

AAA policies are powerful and flexible, as they support a range of authentication and authorization mechanisms that can be “mixed and matched” in a single policy. For example, our AAA policy authenticates the user by validating the signer certificate of the digitally signed message and propagates the requester’s identity to CICS in the form of an ICRX so that it can be mapped to a RACF user ID and used for CICS authorization checking.

The AAA policy editor guides you through the following activities:

- ▶ Defining methods to extract a user’s identity from an incoming request
- ▶ Defining the method to authenticate the user
- ▶ Defining the method to map credentials
- ▶ Defining methods to extract and map resources
- ▶ Defining the method to authorize a request
- ▶ Defining counters for authorized and rejected messages
- ▶ Defining logging for authorizations and rejections
- ▶ Defining post-processing activities

We create the AAA policy using the following steps:

1. Drag and drop the AAA icon to the right of the Verify action on the policy rule (Figure 10-21).

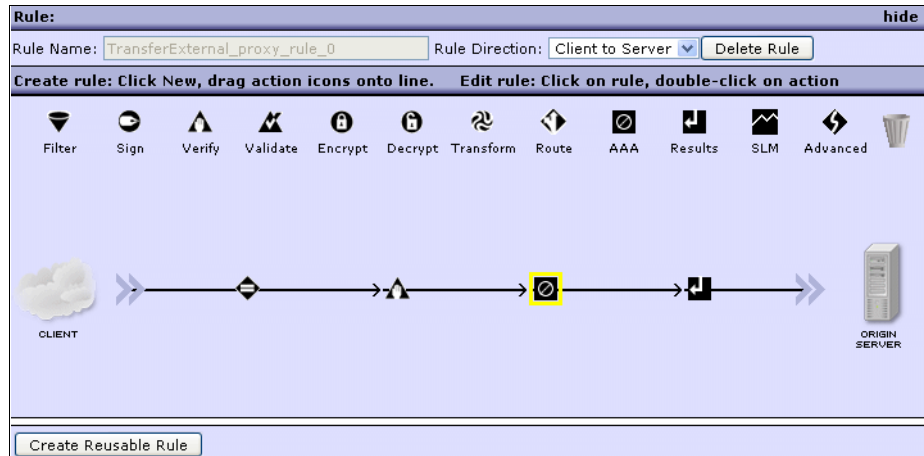


Figure 10-21 DataPower AAA Policy

Double-click the **AAA** icon.

2. On the Configure AAA Action page, click the plus sign (+) to create a new AAA policy.
On the next page (Figure 10-22) specify IdProp as the name of the AAA policy, and then click **Create**.

The screenshot shows the "Configure an Access Control Policy" page in the DataPower configuration interface. The page title is "Configure an Access Control Policy" and it includes a "Help" link. Below the title, there is a section titled "Create a new AAA Policy Object". The "AAA Policy Name" field is set to "IdProp" and has a required field asterisk (*). The "Create" button is highlighted with a red circle, and the "Cancel" button is also visible.

Figure 10-22 DataPower AAA policy 2

3. On the next page (Figure 10-23) check the box **Subject DN from Certificate in the Message's signature** as the method to be used to extract the user's identity from the incoming request, and then click **Next**.

WebSphere. DataPower XI50 IBM

Configure an Access Control Policy [Help](#)

AAA Policy Name: IdProp

Define how to extract a user's identity from an incoming request.

Identification Methods

- HTTP Authentication Header
- Password-carrying UsernameToken Element from WS-Security Header
- Derived-key UsernameToken Element from WS-Security Header
- BinarySecurityToken Element from WS-Security Header
- WS-SecureConversation Identifier
- WS-Trust Base or Supporting Token
- Kerberos AP-REQ from WS-Security Header
- Kerberos AP-REQ from SPNEGO Token
- Subject DN of the SSL Certificate from the Connection Peer
- Name from SAML Attribute Assertion
- Name from SAML Authentication Assertion
- SAML Artifact
- Client IP Address
- Subject DN from Certificate in the Message's signature
- Token Extracted from the Message
- Token Extracted as Cookie Value
- LTPA Token
- Processing Metadata
- Custom Template
- HTML Forms-based Authentication

Validation Credentials for Signing Certificate (none) [v] [+] [...]

Retrieve Remote WS-Sec Token on off

[Back] [**Next**] [Cancel]

Figure 10-23 DataPower AAA Policy 3

- On the next page (Figure 10-24) check the box **Validate the Signer Certificate for a Digitally Signed Message** as the method to be used to authenticate the user. Chose **SmarterBankingShowcaseValidationCred** for validating credential and then click **Next**.

WebSphere. DataPower XI50 IBM

Configure an Access Control Policy Help

AAA Policy Name: IdProp

Define how to authenticate the user.

Method

- Accept a SAML Assertion with a Valid Signature
- Accept an LTPA token
- Bind to Specified LDAP Server
- Contact a SAML Server for a SAML Authentication Statement
- Contact a WS-Trust Server for a WS-Trust Token
- Contact ClearTrust Server
- Contact Netegrity SiteMinder
- Contact NSS for SAF Authentication
- Custom Template
- Pass Identity Token to the Authorize Step
- Retrieve SAML Assertions Corresponding to a SAML Browser Artifact
- Use an Established WS-SecureConversation Security Context
- Use certificate from BinarySecurityToken
- Use DataPower AAA Info File
- Use specified RADIUS Server
- Validate a Kerberos AP-REQ for the Correct Server Principal
- Validate the Signer Certificate for a Digitally Signed Message.**
- Validate the SSL Certificate from the Connection Peer

Signature Validation Credentials

SmarterBankingShowcaseValidationCred + ...

XPath Expression XPath Tool

XPath Bindings

Retrieve Remote WS-Sec Token on off

Define how to map credentials.

Method None *

Back Next Advanced Cancel

Figure 10-24 DataPower AAA Policy 4

- On the next page check the box **URL Sent by Client** as the method to be used to extract the resources, and then click **Next**.
- On the next page check the box **Allow Any Authenticated Client** as the method to be used to authorize a request, and then click **Next**.

- On the next page (Figure 10-25) check the box **Generate an ICRX token for z/OS Identity Propagation** as a post-processing action. Specify DPDev as the ICRX Realm and then click **Commit**.

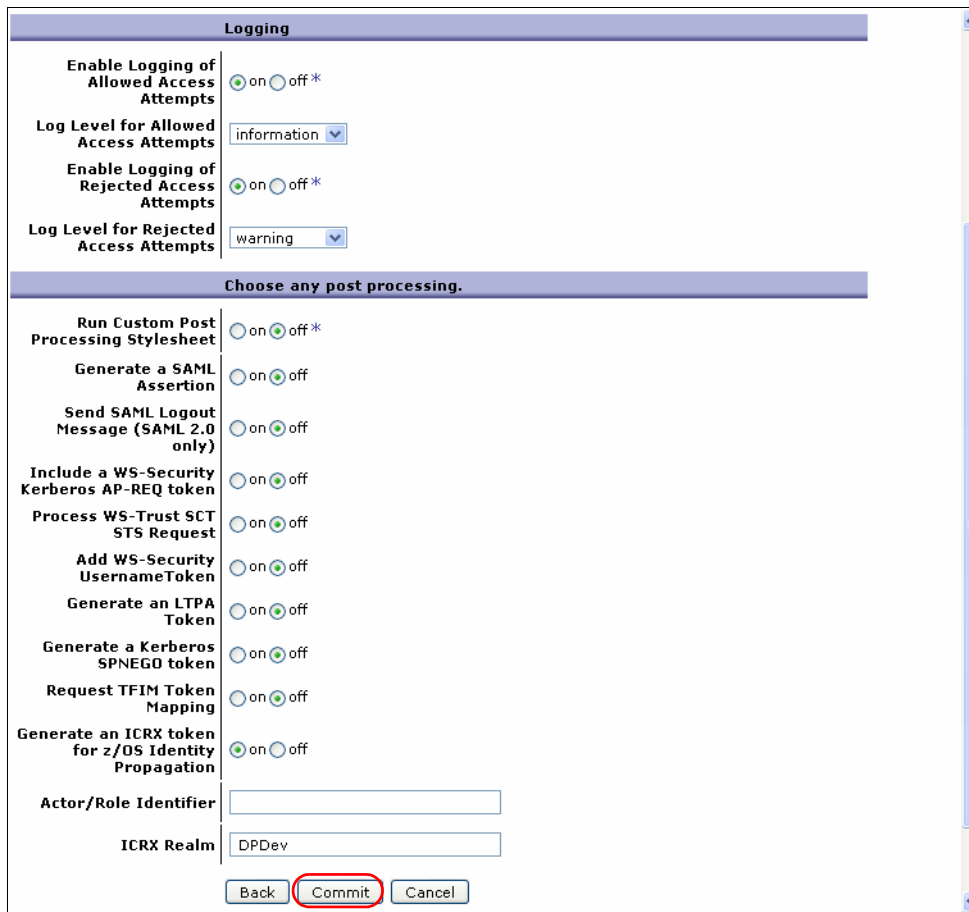


Figure 10-25 DataPower AAA Policy 5

- Click **Done** twice and then **Apply** to finalize the creation of the AAA policy. Figure 10-26 shows the final DataPower policy for the account transfer service.

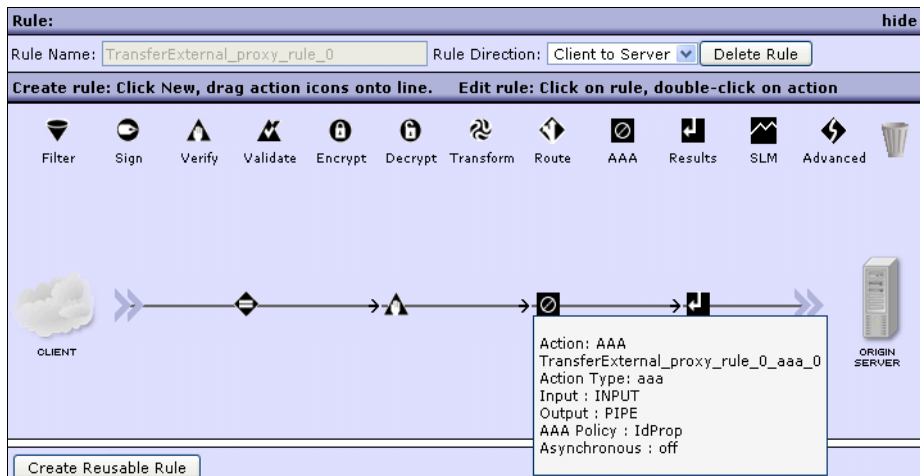


Figure 10-26 DataPower Policy for Account Transfer service

10.6 Configuring CICS

In this section we review the CICS resource definitions that are required to enable the web services identity propagation scenario.

We document the following TOR resource definitions:

- ▶ SIT parameters
- ▶ TCPIPSERVICE
- ▶ WEBSERVICE
- ▶ URIMAP
- ▶ PIPELINE and pipeline configuration file

We document the *MRO connection* Application Owning Region (AOR) resource definition.

10.6.1 SIT parameters

We configured our CICS TOR region with security prefixing, transaction security, and surrogate user security active using the following SIT parameters:

- ▶ SEC=YES to indicate that we want RACF services to control access to CICS resources.
- ▶ XTRAN=YES so that CICS calls RACF to verify that the user ID associated with the transaction is permitted to run the transaction.
- ▶ XUSER=YES to specify that CICS is to perform surrogate user checks.

10.6.2 TCPIPSERVICE

A TCPIPSERVICE definition contains information about the port on which inbound requests are received and whether any transport-based security mechanisms will be applied by CICS.

Example 10-3 shows the TCPIPSERVICE definition that we used in our scenario.

Example 10-3 SSLWEBS TCPIPSERVICE

```
CEDA View TCpipservice( SSLWEBS )
TCpipservice : SSLWEBS
GRoup       : TCPRT11
DEscription : CICS WEB SERVICES
Urm        : DFHWBADX
PORtnumber  : 61001
STatus     : Open
PROtocol   : Http
TRansaction : CWXN
Backlog    : 00005
TSqprefix  :
Host       : ANY
(Mixed Case) :
Ipaddress  :
SOcketclose : No
Maxdatalen : 050000
SECURITY
SSl        : Clientauth
CErtificate : CICSRT10
(Mixed Case) :
PRIVacy    : Supported
```

```

Ciphers      : 0A1613100D05042F30313223
Authenticate : No

Realm       :
(Mixed Case)
Attachsec   :

```

The TCPIPService listens on port number 61001.

- ▶ SSL is set to ClientAuth so that CICS requires connections to this port to provide a client certificate. That is, the DataPower appliance must send its certificate to CICS.

Important: We use SSL client authentication to establish the trust relationship between DataPower and CICS.

- ▶ Ciphers is set to 0A1613100D05042F30313223. This attribute specifies a string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suites that CICS will support for connections to this port.
- ▶ Authenticate is set to No. We are not using the DataPower client certificate to authenticate the request.

Note: We do not document the configuration of SSL between DataPower and CICS in this book. For detailed information about configuring SSL with CICS, see the IBM Redbooks publication *Securing CICS Web Services*, SG24-7658.

10.6.3 URIMAP

A URI mapping or URIMAP resource definition matches the URIs of web service requests. URIMAP definitions for inbound web service requests have a USAGE attribute of PIPELINE. The URIMAP associates a URI for the request with a PIPELINE and WEBSERVICE resource that specifies the processing to be performed.

Importantly, you can use a URIMAP to specify the name of the transaction that CICS uses for running the pipeline alias transaction (the default is CPIH). RACF authorizations can then be set for this transaction.

Example 10-4 shows the URIMAP definition that we used in our scenario.

Example 10-4 UR9ZTE URIMAP

```

CEDA View Urimap( UR9ZTE )
Urimap      : UR9ZTE
Group       : CORPWS2
DEscription : TRANSFER TO EXTERNAL ACCOUNT
STatus      : Enabled          Enabled ! Disabl
USAge       : Pipeline         Server ! Client
UNIVERSAL RESOURCE IDENTIFIER
SCHEME      : HTTP             HTTP ! HTTPS
PORT        : No               No ! 1-65535
HOST        : *
(Mixed Case) :
PATH        : /Showcase/Accounts/TransferExternal
(Mixed Case) :
ASSOCIATED CICS RESOURCES

```

```

TCpipservice   :
ANalyzer       : No
CONverter      :
TRANsaction    : ZTE
PROgram        :
PIPeLine       : PIPFIS
Webservice     : TranExt
ATomService    :
SECURITY ATTRIBUTES
USERid         :
CIPHERs        :
CERTIFICATE    :
AUTHenticate   :

```

Note the following information about Example 10-4 on page 149:

- ▶ This URIMAP will match requests for any host and any TCPIP SERVICE, with a path of /Showcase/Accounts/TransferExternal.
- ▶ The pipeline task will run under transaction ZTE within pipeline PIPFIS.
- ▶ The request is for Webservice TranExt.

10.6.4 PIPELINE

A PIPELINE resource definition provides information about the message handlers that act on a service request and on the response. The information about the message handlers is supplied indirectly. The CONFIGFILE attribute of the PIPELINE definition specifies the name of an HFS file, called the pipeline configuration file, which contains an XML description of the message handlers and their configuration.

Example 10-5 shows the PIPELINE definition that we used in our scenario.

Example 10-5 PIPFIS PIPELINE

```

CEDA View PIPeline( PIPFIS )
  Pipeline       : PIPFIS
  Group          : CORPWS2
  DESCRIPTION    :
  STATUS         : Enabled
  Reswait       : Deft
  CONFIGfile     : /u/cicsuser/services/config/SOAP12provider.xml
  (Mixed Case)  :

```

The path to the pipeline configuration file is:

```
/u/cicsuser/services/config/SOAP12provider.xml
```

Pipeline configuration file

The CICS-supplied message handler that provides support for WS-Security is specified in the web services pipeline configuration file and is configured to allow the pipeline to receive an ICRX.

Example 10-6 shows the pipeline configuration file that we used in our scenario.

Example 10-6 Pipeline configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/software/http/cics/pipeline provider.xsd ">
  <service>
    <service_handler_list>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <authentication trust="blind" mode="basic-ICRX">
            </authentication>
          </dfhwsse_configuration>
        </wsse_handler>
      </service_handler_list>
    </terminal_handler>
    <cics_soap_1.2_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

The `<wsse_handler>` element shown in Example 10-6 contains a `<dfhwsse_configuration>` element that specifies configuration information for the CICS-supplied security handler.

In the `<authentication>` element:

- ▶ The `trust="blind"` attribute specifies that asserted identity is used.
- ▶ The `mode="basic-ICRX"` attribute specifies that inbound messages must contain an ICRX identity token. CICS resolves the identity, puts the user ID in the DFHWS-USERID container, and puts the ICRX in container DFHWS-ICRX.

10.6.5 WEBSERVICE

A WEBSERVICE resource defines the aspects of the runtime environment for a CICS application program deployed as a web service. Three objects define the execution environment that allows a CICS application program to operate as a web service provider:

- ▶ The web service description
- ▶ The web service binding file
- ▶ The pipeline

These three objects are defined to CICS on the following attributes of the WEBSERVICE resource definition:

- ▶ WSDLFILE
- ▶ WSBIND
- ▶ PIPELINE

Note: The WEBSERVICE definition does not have a direct impact on the security context used for processing the request.

Example 10-7 shows the WEBSERVICE definition that we used in our scenario.

Example 10-7 TranExt WEBSERVICE

```
CEDA View Webservice( TranExt )
  Webservice      : TranExt
  Group           : CORPWS2
  Description     :
  Pipeline        : PIPFIS
  Validation      : No
  WSBind          : /u/cicsuser/services/wsbind/CWPEPM01.wsbind
  (Mixed Case)   :
  WSDLfile        : /u/cicsuser/services/wsd1/CWPEPM01.wsd1
  (Mixed Case)   :
```

Note the following information about Example 10-7:

- ▶ The path to the WSBind file is /u/cicsuser/services/wsbind/CWPEPM01.wsbind.
- ▶ The path to the WSDLfile file is /u/cicsuser/services/wsd1/CWPEPM01.wsd1.

10.6.6 MRO connection

After a distributed identity has been mapped to the desired RACF user ID, the distributed identity remains attached to the CICS security context and is automatically transmitted by a distributed program link (DPL) or function shipping requests that flow between interconnected CICS TS 4.1 regions within the same sysplex.

The transmission of distributed identities between CICS regions is dynamic, and all that is necessary is to ensure that the systems are connected via MRO or IPIC connections that specify the propagation of security credentials (this is controlled by the ATTACHSEC or USERAUTH connection parameters, respectively). In addition, the security context can also be transferred between CICS systems in different Sysplexes providing that the systems are connected by a client-authenticated SSL IPIC connection.

In our tested scenario, we used an MRO connection between the TOR and AOR.

Example 10-7 shows the security properties of the MRO connection that we defined for the AOR.

Example 10-8 RT10 CONNECTION

```
CEDA View CONnection( RT10 )
  CONnection      : RT10
  Group           : AORTOR
  ...
SECURITY
  SEcurityname    :
  ATTachsec       : Identify
  BINDPassword    :
  BINDSecurity    : No
  Usedfltuser     : No
```

ATTachsec is set to Identify so that the CICS TOR propagates the RACF user ID and distributed identity to the CICS AOR.

10.7 Configuring RACF

This section reviews the RACF definitions that are required to enable the web services identity propagation scenario. We document the following definitions:

- ▶ Identity mapping rules
- ▶ CICS transaction authorizations

10.7.1 Identity mapping rules

RACMAP commands are used to map the distributed identity to the RACF user ID. The distributed identity DN is stored in the USERDIDFILTER and the realm is the REGISTRY name.

The RACMAP command in our scenario is used to map all distributed identities belonging to business partner 1 to RACF ID PARTNER1, all those belonging to business partner 2 to PARTNER2, and so on. The following sequence of commands is used to define the mapping for business partner 1:

1. Create the new RACMAP identity filter using the MAP function:

```
RACMAP ID(PARTNER1) MAP USERDIDFILTER(NAME('O=IBMPartner1,C=UK'))  
REGISTRY(NAME('DPDev')) WITHLABEL('IBMPartner1')
```
2. Issue the RACF REFRESH command to refresh the IDIDMAP class:

```
SETOPTS RACLIST(IDIDMAP) REFRESH
```
3. List the new RACMAP identity filter using the LISTMAP function (Example 10-9).

Example 10-9 RACMAP LISTMAP command

```
RACMAP ID(PARTNER1) LISTMAP
```

Mapping information for user PARTNER1:

```
Label: IBMPartner1  
Distributed Identity User Name Filter:  
>O=IBMPartner1,C=UK<  
Registry Name:  
>DPDev<
```

Note: The RACF Class IDIDMAP must be activated before you can issue the RACMAP command.

10.7.2 Authorizing the service requester

This section provides the RACF commands that are used to authorize the service requester to run the CICS account transfer web service.

Permitting access

Example 10-11 shows the RACF commands that are used to permit the caller's user ID (PARTNER1) to start the CICS pipeline alias transaction ZTE.

Note: In normal circumstances, we recommend that user groups are granted access to transaction groups. However, for the sake of simplicity, in our example we use a single user and single transaction.

Example 10-10 RACF command to allow PARTNER1 to run ZTE transaction

```
PERMIT ZTE CLASS(TCICSTRN) ID(PARTNER1) ACCESS(READ)
SETROPTS RACLIST(TCICSTRN) REFRESH
```

Permitting surrogate access

If you use identity propagation, it requires that the service provider *trusts* the requester to make the assertion that the propagated user identity is valid. Normally, this means that the original user has already been authenticated.

CICS uses *surrogate security checking* to enable the definition of trust relationships. A surrogate user is a RACF user ID, which is authorized to act on behalf of another user (the original user). CICS uses surrogate user security in a number of different situations, including in the case of web services identity propagation with DataPower.

To enable CICS surrogate user checking, you define the appropriate SURROGAT class profiles for CICS in the RACF database and you authorize CICS surrogate users to the appropriate SURROGAT profiles.

It is important to configure a trust relationship between the DataPower appliance and CICS, for example, using SSL client certification between DataPower and CICS. The digital certificate that DataPower uses to identify itself can be associated with a RACF user ID, and that user ID must be granted surrogate authority to assert the RACF user IDs that are mapped from the propagated distributed identities.

Important: When using identity propagation between DataPower and CICS, you should establish a trust relationship by using RACF SURROGAT class profiles.

Example 10-11 shows the RACF command that would be used to permit a DataPower user ID (DP1) to start pipeline alias transactions with the user ID PARTNER1.

Example 10-11 RACF commands to allow XXXXXX to act as surrogate for PARTNER1

```
RDEFINE SURROGAT PARTNER1.DFHSTART UACC(NONE) OWNER(NIGEL)
PERMIT PARTNER1.DFHSTART CLASS(SURROGAT) ID(DP1) ACCESS(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

Note: We did not enable authentication of the DataPower appliance in our tests.

For more information regarding surrogate user security, see *CICS Transaction Server for z/OS RACF Security Guide*, SC34-6454.

10.8 Testing the scenario

In this section we show the results of testing the scenario. We show the following:

- ▶ Using the Generic Service Client of RDz to invoke the account transfer web service
- ▶ Using the CICS Explorer to view task association data that contains the original distributed identity and the CICS task running under the correct mapped RACF user ID
- ▶ Using the probe facility of DataPower to monitor how DataPower implements identity propagation

We also show the results of the following failure scenarios:

- ▶ No distributed identity (ICRX) is passed by DataPower.
- ▶ A distributed identity (ICRX) is passed by DataPower but no mapping exists for that identity.
- ▶ A mapping exists but the mapped RACF user ID does not have authorization to run the CICS transaction.

10.8.1 Successful many-to-one identity mapping

This section shows the transaction flow of a normal successful request to run the CICS account transfer service.

Invoking the web service

In 10.4, “Configuring RDz” on page 129 we show how the Generic Service Client of RDz can be used to test web service invocations.

Figure 10-3 on page 130 shows the successful invocation of the account transfer service.

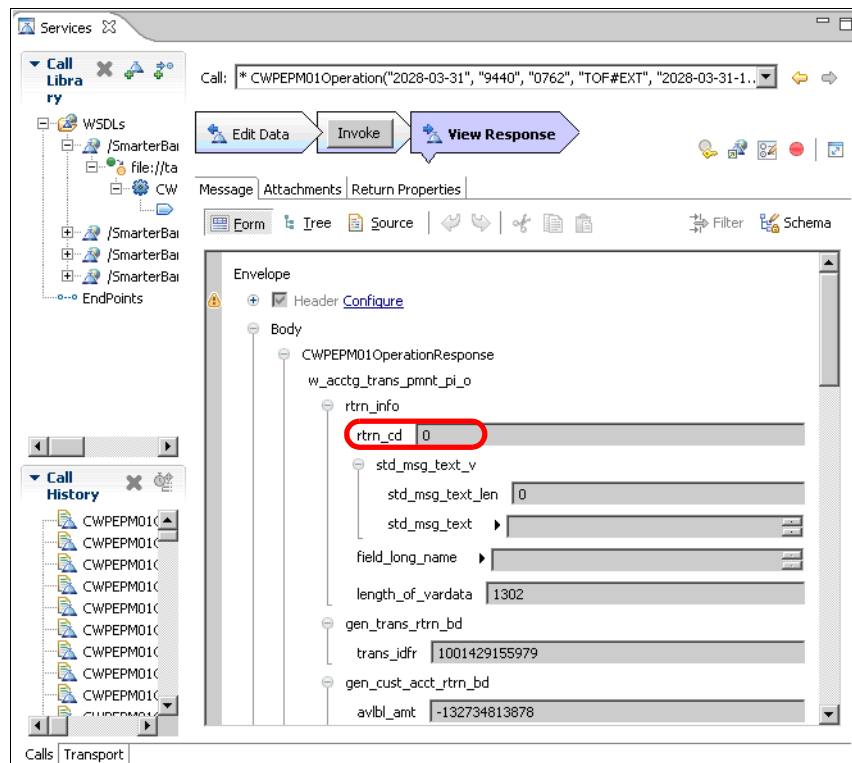


Figure 10-27 Successful invocation of the account transfer service

The return code for the request is 0, which indicates that the transaction was successful.

Task association data

CICS association data is a set of information that describes the environment in which CICS tasks run and the way that tasks are attached in a CICS region. Some association data is specific to the task itself, for example, the task ID and the user ID relating to the task.

You can use the CICS Explorer to view task association data. Figure 10-28 shows a ZTE task running with the user ID PARTNER1 (the user ID that has been mapped from the distributed identity in the ICRX). It also shows the distributed identity (DN) CN=Bob Clark, OU=Retail, O=IBMPartner1, C=UK.

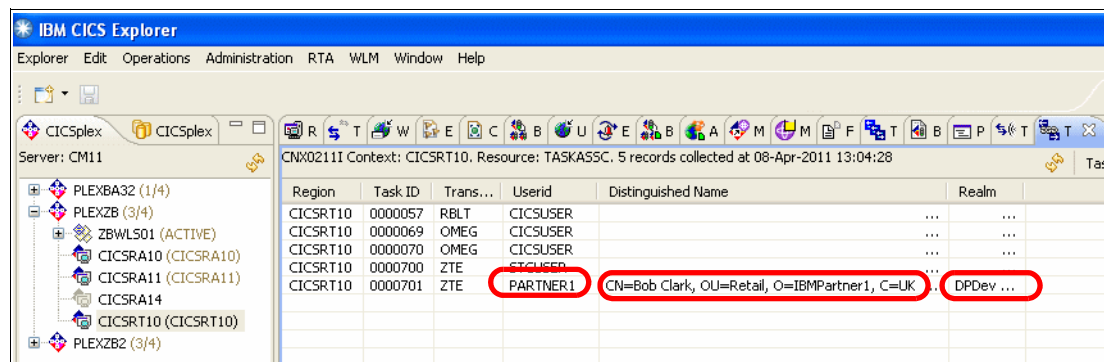


Figure 10-28 CICS task association data

Using the probe facility of DataPower

The DataPower multi-step probe provides a very powerful feature to perform debugging and effective problem determination. We use the probe to view the message as it passes through the appliance.

On the Configure Web Service Proxy page (Figure 10-7 on page 134), we click the **Show Probe** action for the TransferExternal_proxy web service proxy. We then click **Enable Probe** (Figure 10-29).

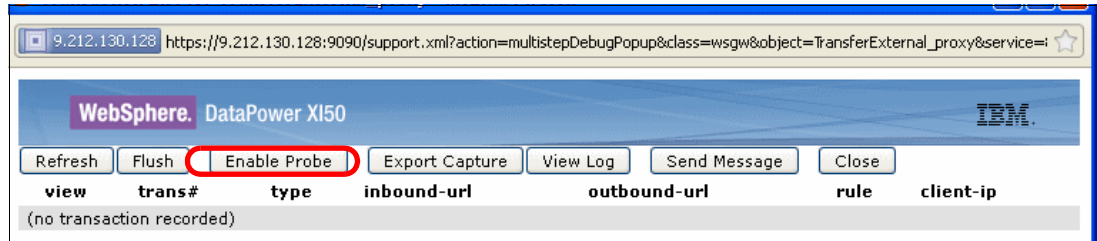


Figure 10-29 DataPower probe

After enabling the probe, we invoke the web service and refresh the list of transactions that have been captured by the probe (Figure 10-30).

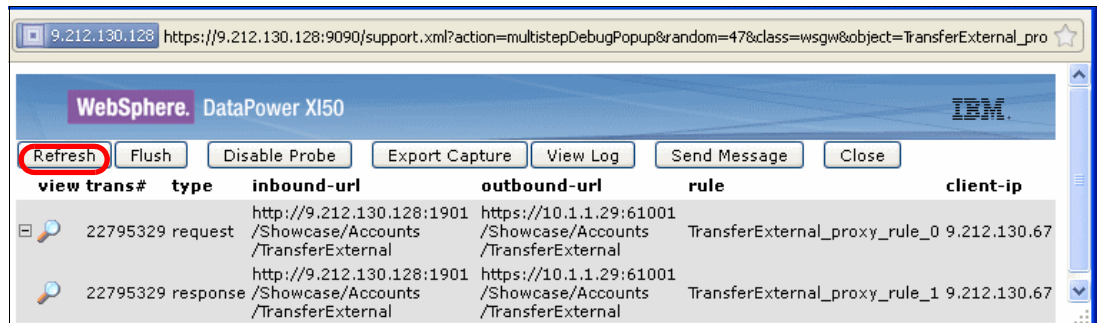


Figure 10-30 DataPower Probe 2

The probe shows the request and response messages for the account transfer web service. We click the eyeglass of the request to view the details of the message. Figure 10-31 shows the signed message received by DataPower. We see the BinarySecurityToken that contains Bob's X.509 certificate and signature parts of the Security SOAP header.

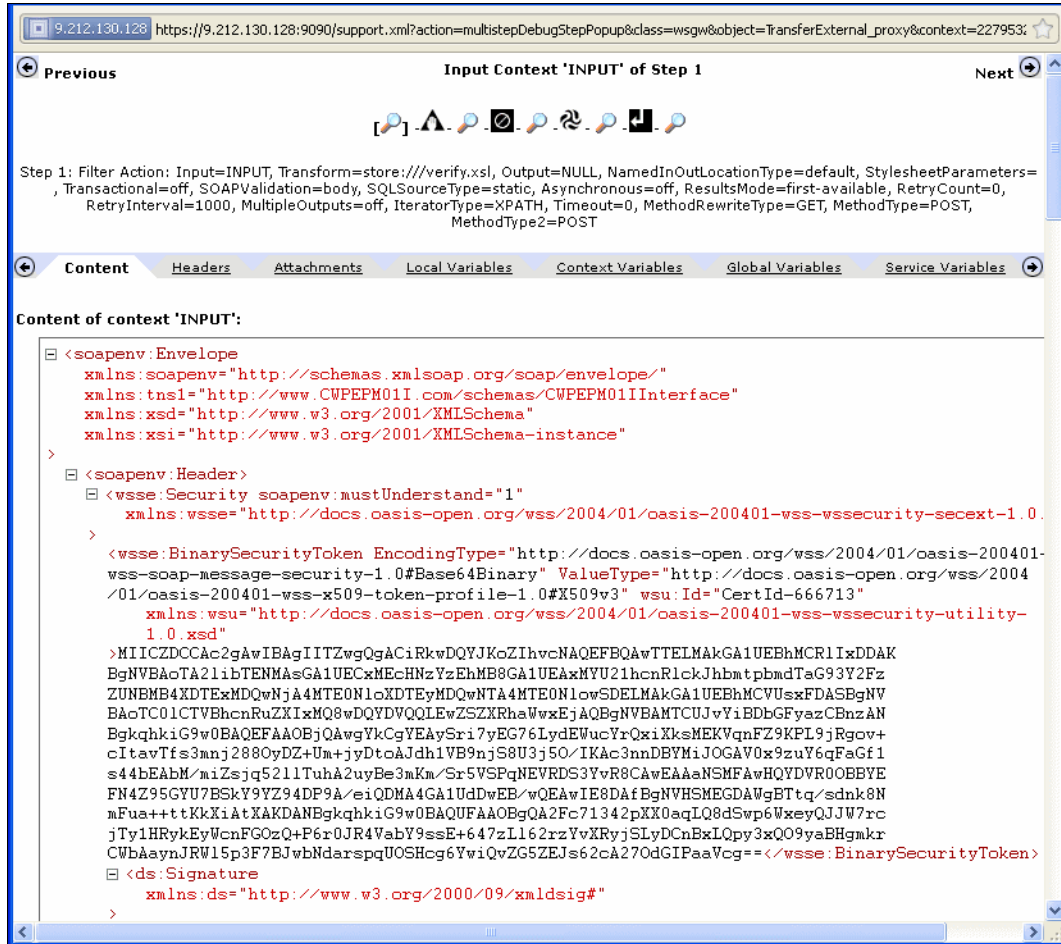


Figure 10-31 DataPower Probe 3

Using the DataPower probe we can view the message as it is processed by the actions that we have configured in DataPower for the account transfer service. For example, Figure 10-32 shows the message after running the AAA policy IdProp.

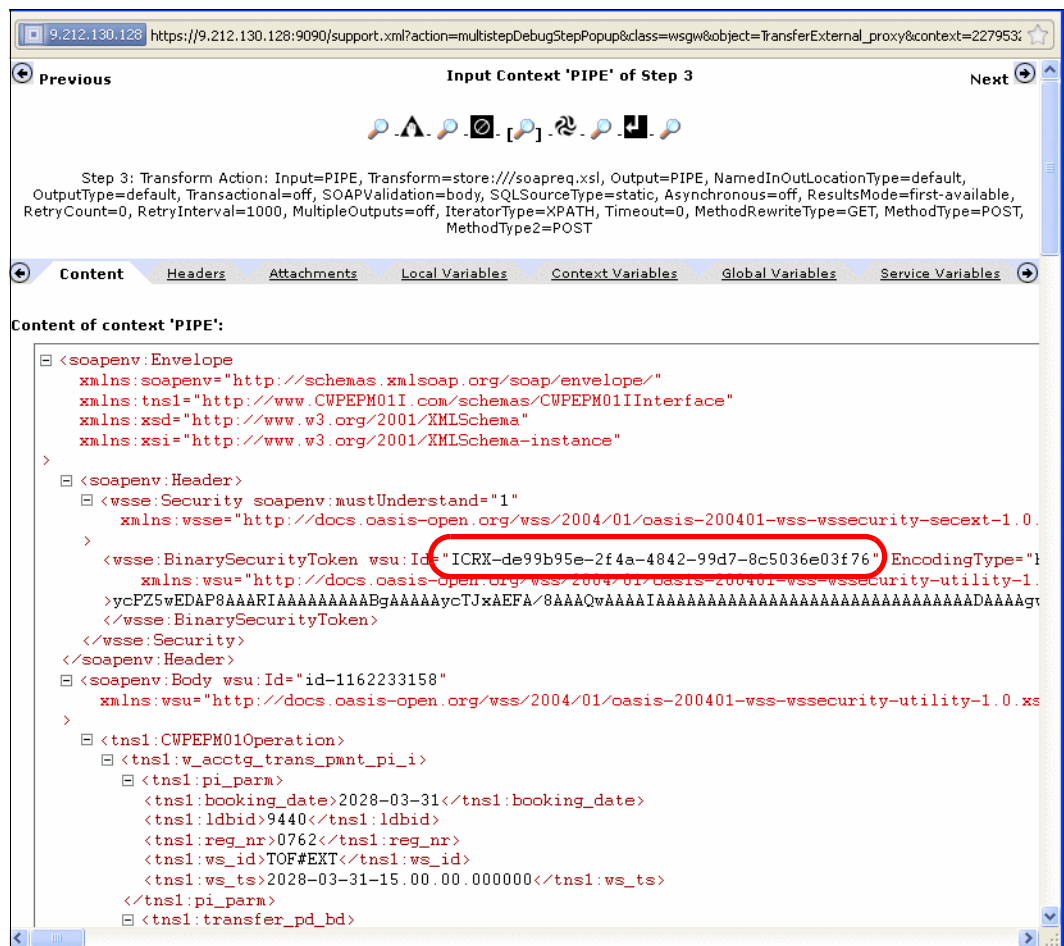


Figure 10-32 DataPower Probe 4

After the AAA policy has executed, we see that the request message still contains a BinarySecurityToken, but now it is in the form of an ICRX token. This ICRX token contains the distinguished name of Bob. The signature part of the message has been removed.

Note: For more information about ICRX identity tokens, see *z/OS Security Server RACF Data Areas*, SA22-7680.

10.8.2 Failure scenarios

This sections discusses SOAP faults and error messages that you might see when testing identity propagation with CICS Web services.

No distributed Identity (ICRX) is passed by DataPower

CICS expects to receive an ICRX in the SOAP request message, but DataPower does not send an ICRX.

The SOAP fault for this message (Example 10-12) highlights that CICS cannot authenticate the service requester.

Example 10-12 SOAP fault - ICRX not sent

```
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    <faultcode>wsse:FailedAuthentication</faultcode>
    <faultstring>A security token could not be authenticated</faultstring>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
```

No mapping exists for distributed identity

CICS receives an ICRX in the SOAP request message, but no mapping has been defined to map the distributed identity to a RACF user ID.

The SOAP fault for this message is the same as the one shown in Example 10-12. In addition, a RACF message is issued as shown in Example 10-13.

Example 10-13 RACF message - no mapping

```
ICH408I USER(CICSUSER) GROUP(SYS1) NAME(CICSUSER)
  DISTRIBUTED IDENTITY IS NOT DEFINED:
  CN=Jessica Jones, OU=Miami Branch, O=IBMPartner3, C=US DPDev
```

No authorization to run the CICS transaction

CICS receives an ICRX in the SOAP request message, and a mapping has been defined to map the distributed identity to a RACF user ID, but the mapped RACF user ID is not authorized to run the CICS transaction.

Example 10-14 shows the RACF error message that is written to the system log if the user is not authorized to run the CICS service provider application.

Example 10-14 RACF message - unauthorized access to CICS transaction

```
ICH408I USER(PARTNER2) GROUP(SYS1 ) NAME(PARTNER2)
  ZTE CL(TCICSTRN)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )
```

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.

- ▶ *Implementing CICS Web Services*, SG24-7206
- ▶ *DB2 10 for z/OS Technical Overview*, SG24-7892
- ▶ *Securing CICS Web Services*, SG24-7658

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS Security Server RACROUTE Macro Reference*, SA22-7692
- ▶ *z/OS Security Server RACF callable Services*, SA22-7691
- ▶ *z/OS Security Server RACF Data Areas*, SA22-7680
- ▶ *z/OS Security Server RACF macros and Interfaces*, SA22-7682
- ▶ *CICS Transaction Gateway z/OS Administration*, SC34-7058-02

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

Accessor Environment Element 81
ACEE 81
Authenticate 149
authentication registry or Realm 4

C

CICS ECI (External Call Interface) resource adapter 26
CICS resources that a systems programmer uses to configure CICS Web services which are accessed over HTTP 30
CICS resources that a systems programmer uses to configure z/OS identity propagation with the CICS TG 29
CICS Transaction Gateway (CICS TG) 26
CICS Web Services 29
Ciphers 149
communication vector table (CVT) 78

D

data areas impacted by Identity Propagation 78
DB2 10 for z/OS Audit Policy 37
DB2 for z/OS exploitation of Identity Propagation 34
DB2 restrictions on distinguished name 36
DB2 role 36
default RACMAP filter 46
distinguished name explanation 9
Distinguished Name, 4
distributed identify filter 40
Distributed Identity 3
 components of 4
Distributed Identity Data (IDID) 83
Distributed Identity Data Structure (IDID) 11
Dynamic Statement Cache 38

E

ENF event code 71 83
ENVR object 16
Event Notification Facility (ENF) event 71 16
Extended Identity Context Reference (ICRX) 12, 82

F

filter management 50
filter management examples 52

G

General Resource Distributed Identity Mapping Data Record 18

H

HFS
 file 150

how changes to a user RACF profile is handled when there exists a cached identity context for a user 16
how DataPower can be used to propagate a distributed identity to CICS so that it can be mapped to a RACF id 32
how RACF matches the filter value 50

I

ICH04018I message when attempting to delete a RACF user when an identity mapping profile exists for user 19
ICH408I message when distributed identity not found 19
ICHPRCVT 78
ICRX 13, 82
Identity Propagation
 enhanced audit trail 3
 mapping process 4
 RACMAP 4
 user ID Filters 4
 overview 2
IDID 83
IDIDMAP class 43
IDIDMAP profile name (derived from the user name) and the registry name are encoded as UTF-8 data 44
IKJT000 updates 22
initACEE 79
IPCONN 29
IRRRID00 (Remove ID Utility) 20

L

local identity context cache 16

M

Many-to-one filter management 51

N

new messages related to the RACMAP command 47

O

One-to-one filter management 50
overview of a CICS TG identity propagation scenario 28

P

PIPELINE 30, 150
pipeline alias transaction 149
Portnumber 149

R

R_cacheserv 16, 80
R_usermap 81
RACF General template updates for identity propagation 20

RACF settings to permit data records to be written to SMF for identity propagation 57

- AUDIT 57
- CMDVIOL 57
- SAUDIT 57

RACF user profile can not be deleted when an identity mapping exists for user profile 19

RACF User template updates for identity propagation 19

RACMAP command 40

RACMAP command are eligible for automatic direction of application updates 46

RACMAP command authorization 40

RACMAP command parameters 41

- ID(userid) 41
- MAP 41
- REGISTRY 41
- USERDIDFILTER 41

RACMAP command syntax 41

RACMAP DELMAP commands are generated by RACF Remove ID Utility 20

RACMAP implications for RACF user profiles 45

RACROUTE REQUEST=VERIFY/VERIFYX 78

Redbooks Web site 161

- Contact us xi

Registry name required by DB2 36

Registry value 10

Relative Distinguished Names 4

- mapping process is sensitive to value of 5
- parsing during mapping process 4
- value is case sensitive 4

reporting audit information from SMF data 60

- from DB2 tables 63
- SMF UNLOAD producing XML data 61
- SMF Unload Utility 60
- using DFSORT 64
- using IBM Security zSecure Admin 68

retrieve a Distributed Identity and Registry by CICS program 5

RRSF 46

S

SAF calls 13

- initACEE 14
- RACROUTE REQUEST=VERIFY 15

SAF Interfaces and Identity Propagation

- initACEE 12
- R_cacheserv 13
- RACROUTE REQUEST=VERIFY 12

sample members to take unloaded SMF data and then load into DB2 tables 22

SIT parameters

- XTRAN=YES 148

SMF data captured by RACMAP create or delete 56

SMF data captured for RACMAP command which creates or deletes a IDIDMAP profile 56

SMF data captured for RACROUTE with VERIFY or VERIFYX 56

SMF records and Identity Propagation 57

- Type 80 record 58
- Type 83 record 59

SMF reporting for DB2 37

SMF Unload Utility 59

SOAP message security 31

SPTGK00715 48

SPTJJ00026 45

SPTOB39953 47

SPTOB39954 47

SPTOB39955 47

SPTOB39957 47

SPTOB39958 47

SPTOB39960 47

SPTOB39961 48

SPTOB39962 48

SPTOB39963 48

SPTOB39964 48

SPTOB39965 48

summary of z/OS Identity Propagation steps 8

surrogate access 154

T

TCPIPSERVICE 29–30

Transport based security 31

trusted connection 35

types of security mechanisms in a CICS Web services environment 31

U

updating a distributed identity filter 44

URIMAP 30, 149

User Associated Distributed Mapping Record 18

UTF-8 data 43

W

Web service 151

WEBSERVICE 30, 149, 151

- attributes

 - WSBIND 151

 - WSDLFILE 151

WebSphere Application Server 90

WebSphere DataPower 31

- Configure Crypto Certificate 136

- Configure the Crypto Validation Credentials 137

WebSphere DataPower in conjunction with CICS Web services 32

Z

z/OS Identity Propagation conceptual overview 8

z/OS Identity Propagation steps 8

z/OS Security Server events captured by SMF related to Identity Propagation 56

z/OS System Management Facility (SMF) 55



z/OS Identity Propagation



Redbooks®

Scenarios using CICS TG, DB2, and CICS Web services

This IBM Redbooks publication explores various implementations of z/OS Identity Propagation where the distributed identity of an end user is passed to z/OS and used to map to a RACF user ID, and any related events in the audit trail from RACF show both RACF and distributed identities.

How to use SMF records for audit reporting

This book describes the concept of identity propagation and how it can address the end-to-end accountability issue of many customers. This book describes, at a high level, what identity propagation is, and why it is important to us. This book shows a conceptual view of the key elements necessary to accomplish this.

Technical overview of identity propagation

This book provides details on the RACMAP function, filter management, and how to use the SMF records to provide an audit trail. In-depth coverage is provided about the internal implementation of identity propagation, such as information about available callable services.

This book examines the current exploiters of z/OS Identity Propagation and provide detailed examples covering CICS with CICS Transaction Gateway, DB2, and CICS Web services with Datapower.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7850-00

ISBN 0738436062