

RACF for WebSphere on z/OS

Theo Antoff, antoff IT inc, September 2005

RACF for WebSphere on z/OS	1
1 WebSphere Infrastructure.....	2
1.1.Base application server cell.....	2
Daemon	2
Application Server.....	2
1.2.Network Deployment cell.....	3
1.3.Federation.....	3
2 Setting up security	4
2.1 User identification and authentication in WebSphere.....	5
2.2 Authorization checking.....	6
3 ISPF Customization Dialogs.....	6
4 RACF Environment for WebSphere.....	7
4.1 Setting up RACF controls for USS.....	7
z/OS UNIX level security.....	7
4.2.RACF groups for WebSphere.....	8
WebSphere system programmers group	8
WebSphere configuration group.....	8
Servants group.....	9
Default Group for Machine User Ids used by USS.....	9
WebSphere application administrators group.....	9
4.3.RACF user IDs for WebSphere	9
Daemon started task user ID.....	9
Controller started task user ID.....	10
Servant started task user ID	11
CTRACE started task user ID.....	11
WAS Administrator user ID	11
Unauthenticated user ID	12
4.4.Data set profiles for WebSphere.....	12
WebSphere data sets for ISPF dialogs.....	12
WebSphere HFS data sets	12
WebSphere CTRACE data sets	13
WebSphere data set for RACF commands	13
WebSphere log stream data sets	13
RRS data sets.....	13
4.5.Profile in class JESSPOOL	13
4.6. Profiles in class STARTED.....	14
Profiles for daemon, controller, and servant procedures	14
Profile for CTRACE procedure.....	14
4.7.Profiles in class APPL.....	14
4.8.Profiles in class LOGSTRM.....	14
4.9.Profiles in class FACILITY.....	15
Protect LOGR policy.....	15
Protect Coupling Facility Structure	15
Access to WLM policy	15
FASTPATH processing.....	15
Access to Commands LIST Digital Certificates and LIST Key Rings	15
Access to IMS	16
4.10.Profiles in class CBIND	16
4.11.Profiles in class SERVER.....	17
4.12 Setting up SURROGAT check in CICS for remote program	17
4.13.Profile in class OPERCMDS.....	17
4.14.Profiles in class EJBROLE.....	18
4.15.Profiles in class CSFSERV.....	19

4.16.RACF protection for connection of WebSphere to DB2.....	19
Profile in class DSNR.....	19
Secondary authorization IDs.....	20
DB2 GRANT Statements	20
4.17. Setting permission for HFS files created by applications	20
5 Setting up SSL security for WebSphere for z/OS.....	20
5.1.SSL Objectives.....	20
5.2 SSL client certificate security for your WebSphere Application Server and clients.....	22
5.3 Defining SSL security for servers and clients.....	23
Using RACF to authorize the server to use digital certificates	23
Setting up SSL security for clients.....	24
RACF authorizations for SSL.....	24
Mapping client digital certificates to MVS user IDs	25
5.4 Using certificates to set up HTTPS internal transport connections	25
Setting up secure HTTPS internal transport connections using server certificates signed by an internal CA	26
Setting up secure HTTPS internal transport connections using client certificates signed by an internal CA	28
Setting up secure HTTPS internal transport connections using server certificates signed by an external CA.....	30
Setting up secure HTTPS internal transport connections using client certificates signed by an external CA.....	32
6 CLIST to define RACF controls for WebSphere.....	33

This article provides a comprehensive description of all necessary controls and techniques to establish security for WebSphere on the zSeries Operating system (z/OS). We discuss all RACF groups, user IDs, classes and profiles needed to protect WebSphere Version 5 Release 1.

1 WebSphere Infrastructure

Implementing security in a WebSphere environment requires that security administration, systems programming, and WebSphere application development staffs work closely together. Terminology can sometimes become a problem when these disparate groups exchange information. In this section, we present an overview of WebSphere infrastructure and terminology.

1.1.Base application server cell

The **Base application server cell** is the simplest operating structure in WebSphere for z/OS. It includes one **Cell** with a Location service daemon (or simply **Daemon**) and one **Node** which includes an **Application Server**.

Daemon

The daemon server is a special server running as a started task. It is responsible for accepting and initiating application requests. For that it has to know which Application servers are active and know all the applications in them. There is only one daemon per cell per LPAR needed in the architecture of WebSphere Application Server .

Application Server

The Application Server consists of one **Controller**, one or more **Servants and J2EE** (Java2 Enterprise Edition) **server**. The J2EE server includes **Web container**, **EJB** (Enterprise Java Beans) **container**, **HTTP** (HyperText Transport Protocol) **internal transport** and **JMS** (Java Message Server) **server**.

The Controller and the Servants run as started tasks. The Java Virtual Machine (JVM) resides in the servant address space. The controller and servant structure design makes it possible to

start multiple servants by Work Load Manager (WLM), based on the workload queued by the controller region.

1.2. Network Deployment cell

The **Network Deployment cell** or **Deployment Manager Node** (after **Federation**) provides more robust operating environment with the unique ability to:

- Span multiple LPARs in a sysplex
- Fully utilize **Clustering**
- Manage Application servers and applications from **Administrative console**

The Deployment Manager is the sole occupant of its own node structure, there are no Application servers in the node, and a cell can may have only one Deployment Manager. Again, the Deployment Manager node includes Daemon, Controller and any number of Servants.

1.3. Federation

Federation is the job of coupling Base Application Server cells onto the Deployment Manager Cell. After executing this job, only one Cell and one Daemon are left – those of the Deployment Manager. The cell of the Base Application server and its Deamon are dropped. The Application Server after Federation includes a new started task, called **Node Agent**. It provides node level administrative function .

Here is a quick breakdown of clusters, nodes and cells:

Cluster	A logical collection of like-configured Application servers (cloned from the administrative console) .Clustering is typically applied to a multi-node cell, where each node is configured on a separate LPAR and the cluster has a member (Application server) at each node.
Node	A logical collection of managed servers on a particular LPAR in the cell. A node can contain servers that are part of clusters that span other nodes, but the node itself is confined to a single LPAR and a single cell. Multiple nodes can exist in the same cell. Each node, managed by a Deployment manager must have a node agent.
Cell	A logical collection of nodes that makes up an administrative domain or boundary. The nodes that comprise a cell must be configured on LPARs in the same sysplex. We can have multiple cells in the same sysplex managed by Deployment Manager or not managed (stand-alone as Base cells). The cell is the largest unit of organization in Websphere V5.

For those who wish to delve into more details of how RACF and J2EE interact, we recommend the following manuals and Redbooks:

WebSphere® Application Server for z/OS, V5.1, Getting Started,GA22-7957-02
ftp://ftp.software.ibm.com/software/webserver/appserv/zos_os390/v51/bbo5a102.pdf

WebSphere® Application Server for z/OS, V5.1, Security,SA22-7961-02
ftp://ftp.software.ibm.com/software/webserver/appserv/zos_os390/v51/bbo5e102.pdf

Wunderlich, H et al, z/OS WebSphere Application Server V5 and J2EE 1.3 Security Handbook
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246086.pdf>

2 Setting up security

WebSphere for z/OS supports access to resources by clients and clusters in a distributed network, so part of your security strategy should be to determine how to control access to these resources and prevent inadvertent or malicious destruction of the system or data. These are the pieces in the distributed network that you must consider:

- Authentication is the process of establishing whether a client is valid in a particular context. A client can be either an end user, a machine, or an application. An authentication mechanism in WebSphere Application Server typically collaborates closely with a User Registry. When configuring a cell, you must select a single authentication mechanism. The choices for authentication mechanism include:
 - Simple WebSphere Authorization Mechanism (SWAM) — only on Base Application Server, not available on the Network Distributed configuration
 - Lightweight Third Party Authentication (LTPA)
 - Integrated Cryptographic Service Facility (ICSF)
- You must authorize clusters to the base operating system services in z/OS. These services include SAF security, database management, and transaction management.
 - For the clusters, you must distinguish between controllers and servants. Controllers run authorized system code, so they are trusted. Servants run application code and are given access to resources, so you should carefully consider the authorizations you give servants.
 - You must also distinguish between the level of authority run-time clusters and your own application clusters have. For example, the node needs the authority to start other clusters, while your own application clusters do not need this authority.
- You must authorize clients (users) to clusters and objects within clusters. The characteristics of each client requires special consideration:
 - Is the client on the local system or is it remote? The security of the network becomes a consideration for remote clients.
 - Will you allow unidentified (unauthenticated) clients to access the system? Some resources on your system may be intended for public access, while others need to be protected. In order to access protected resources, clients must establish their identities and have authorization to use those resources.
- Information about users and groups reside in a user registry. In WebSphere Application Server, a user registry authenticates a user and retrieves information about users and groups to perform security-related functions, including authentication and authorization. Implementation is provided to support multiple operating system or operating environment based user registries. When configuring a cell, you must select a single user registry. The user registry can be local or remote. The choices for user registry include:
 - SAF-based local registry (default)
 - Lightweight Directory Access Protocol (LDAP) — LDAP can be either a local or remote registry
 - Custom user registry — a Custom Registry is provided by 3rd party. WebSphere provides a simple user registry sample called the FileBasedRegistrySample.

If you need to protect resources, identifying who accesses those resources is critical. Thus, any security system requires client (user) identification and authentication. In a distributed network supported by WebSphere for z/OS, clients can be accessing resources from:

- Within the same sysplex as the cluster.
- Remote z/OS systems.
- Heterogeneous systems, such as WebSphere on distributed platforms, CICS, or other Java-compliant systems.

Additionally, clients may request a service that requires a cluster to forward the request to another cluster. In such cases, the system must handle delegation, the availability of the client identity for use by intermediate clusters and target clusters.

Finally, in a distributed network, how do you ensure that messages being passed are confidential and have not been tampered? How do you ensure that clients are who they claim

to be? How do you map network identities to z/OS identities? These issues are addressed by the following support in WebSphere for z/OS:

- The use of SSL and digital certificates.
- Kerberos.
- Common Secure Interoperability Version 2 (CSlv2).

2.1 User identification and authentication in WebSphere

Proper security for any system requires that users or programs identify themselves and prove they are who they claim to be (authenticate themselves). WebSphere for z/OS uses within and across systems the following kinds of user authentication :

1. Local clients and clusters use their user IDs to identify themselves when requesting a service. WebSphere for z/OS uses a transportable form of the user's Accessor Environment Element (ACEE), called a RACO, for local clients and clusters running in the same sysplex. The RACO is used throughout the WebSphere for z/OS system and ensures that any task is performed under the requestor's identity. No authentication is required because the user's identity is already established by the operating system. Just like other z/OS applications, WebSphere for z/OS uses the operating system to keep track of the user identities and makes calls to the security service during the execution of a piece of work.
2. Unless you can be sure all messages exchanged flow exclusively within a trusted network, authenticity of clients and clusters, message confidentiality, and message integrity become important issues. A client may want to be sure that it is receiving a service from a legitimate cluster and a cluster may want to be sure who the client is. Each party also wants to be sure that messages exchanged are protected from tampering or snooping by a malicious third party, so security in the transportation medium (message protection) is a concern. WebSphere for z/OS provides several authentication mechanisms, some of which involve message protection. You need to decide, based on the nature of your network, which authentication mechanism you need. WebSphere Application Server for z/OS V5 supports the following authentication mechanisms:
 - The Simple WebSphere authentication mechanism (SWAM) is intended for simple, non-distributed, single application server run-time environments. The single application server restriction is due to the fact that SWAM does not support forwardable credentials. If a servlet or enterprise bean in application server process 1, invokes a remote method on an enterprise bean living in another application server process 2, the identity of the caller identity in process 1 is not transmitted to server process 2. What is transmitted is an unauthenticated credential, which, depending on the security permissions configured on the EJB methods, can cause authorization failures. Since SWAM is intended for a single application server process, single signon (SSO) is not supported. The SWAM authentication mechanism is suitable for simple environments, software development environments, or other environments that do not require a distributed security solution.
 - Lightweight Third Party Authentication (LTPA) is intended for distributed, multiple application server and machine environments. It supports forwardable credentials and single signon (SSO). LTPA can support security in a distributed environment through cryptography. This supports permits LTPA to encrypt, digitally sign, and securely transmit authentication-related data, and later decrypt and verify the signature. The Lightweight Third Party Authentication (LTPA) protocol enables the WebSphere Application Server to provide security in a distributed environment using cryptography. If you need to interoperate with network distributed servers, you will need to use LTPA.
 - Integrated Cryptographic Service Facility (ICSF) uses hardware encryption available on zSeries processors. ICSF also generates security tokens for authenticated users which can be propagated over to other servers. The main advantage of ICSF is that the keys are stored in secure hardware and only the key label is provided.

3. Within the sysplex, all security protocols (except for RACO) are supported between clients and clusters within the sysplex. Additionally, PassTickets are supported, in which the client's user ID is used for identification and a PassTicket for authentication. A PasTicket is a one-time-use password that is dynamically generated. Because communications within a sysplex flow directly over a protected network, WebSphere for z/OS avoids the overhead of message encryption for these communications. In other words, when systems in a sysplex are directly connected, WebSphere for z/OS determines that the communication is guaranteed to be secure, and does not use encryption. When a client connects to a cluster, part of the connection includes a negotiation between the client and cluster about what security protocol is to be used.

2.2 Authorization checking

When a request flows from a client to the cluster or from a cluster to a cluster, WebSphere for z/OS passes the user identity (client or cluster) with the request. Thus each request is performed on behalf of the user identity and the system checks to see if the user identity has the authority to make such a request. Resource managers such as DB2, IMS, and CICS have implemented their own resource controls, which control the ability of clients or clusters to access resources.

3 ISPF Customization Dialogs

The installation of WebSphere comes with ISPF customization dialogs in which **variables** representing names of RRS and IXGLOGR data sets, CTRACE data sets, HFS data set, short and long names for HFS directories, job names, names of PROCs for SYS1.PROCLIB, TCP/IP ports, RACF userids, and groups, etc should be supplied by WebSphere sysprogs (it is supposed the RACF entities to be advised by the shop's RACF security group).

The dialogs produce REXX EXECs in an output data set containing data, for example:

HLQ.WAS.V510.CELL1.BASE.**DATA**

and JCLs in an output data set, for example:

HLQ.WAS.V510.CELL1.BASE.**CNTL**.

The variables are saved to and loaded from another data set, for example:

HLQ.WAS.V510.CELL1.BASE.**SAVECFG**.

Attention: You should always save your variables into a data set of the type of SAVECFG. You have to always LOAD your variables (by pressing L in the ISPF dialogs menu) before generating a new set of CNTL and DATA data sets.

Note. All data sets names and General Resource names and their profiles in this article are examples ONLY. You are free to name any data sets related to WebSphere according to your shop naming standards. If you do not use a common High Level Qualifier (HLQ) for customized in-house software, then we recommend to use WAS.

As part of the ISPF customization dialog, a REXX EXEC called BBOWBRAC residing in the **DATA** data set is fed with RACF variables such as userids, groups, UIDs and GIDs, which then are wrapped into RACF commands generated into another dynamically built REXX EXEC called BBOWBRAK. The latter is executed with a job BBOCBRAK from the **CNTL** data set.

Although BBOWBRAC is a valuable tool, we undertook a different approach in our WebSphere installation, based on two assumptions:

- Each shop has its own sets of naming standards, procedures and RACF design style already in place, which may not accommodate the BBOWBRAC-generated RACF commands.
- Achieve full separation of duties: RACF environments must be set up by RACF security group and system programming work must be done by WebSphere system programmers.

We created a TSO job as member **WASRACF** in dataset HLQ.WAS.RACF to execute all commands to establish the RACF environment for WebSphere. The commands from this job are explained in the subsequent paragraphs.

Our strategy was to keep the creation of groups, userids, started tasks and dataset profiles to an absolute minimum. If your shop uses a shared RACF data base for DEV/TEST/PROD, then you must use three sets of naming standards (with prefix or suffix D,T and P, for Development, Test and Production, respectively).

WebSphere systems programmers authorized with access CONTROL to profile SUPERUSER.FILESYS.** in class UNIXPRIV should run all ISPF customization jobs (except the RACF generated one, BBOWBRAC).

User IDs from the RACF security group with SYSTEM SPECIAL should run the WASRACF job. They do not need to have UID=0, but should be authorized with access CONTROL to profile SUPERUSER.FILESYS.** in class UNIXPRIV.

4 RACF Environment for WebSphere

This chapter assumes the readers who plan to use z/OS WebSphere already have a RACF environment established for UNIX System Services (USS). See **z/OS UNIX System Services Planning , GA22-7800**.

4.1 Setting up RACF controls for USS

Although there are no prerequisite products for WebSphere Version 51, you may need RACF environments for z/OS UNIX level security, Public Key Infrastructure Services (PKI Services), Integrated Cryptographic Services Facility (ICSF), front end z/OS HTTP server, Lightweight Directory Access Protocol (LDAP) server and DB2.

z/OS UNIX level security

More details about the two possible levels of security (UNIX level of security and z/OS level of security) can be found in Chapter 27 of **z/OS UNIX System Services Planning, GA22-7800** or in <http://www.antoff-it.com/unix-security.pdf>.

The RACF controls necessary to establish the z/OS UNIX level of security are: **Program control** and **Daemon and server control**.

Program control

Library BBO.SBBOLOAD contains the WebSphere programs and it must be ADDMEMed to profile ** in class PROGRAM if this library is not to be loaded into LPA.. Library SGSKLOAD contains the system SSL programs. It also must be ADDMEMed to ** PROGRAM and also added to LNKLIST.

```
RALT PROGRAM ** ADDMEM(' +  
'WAS.V500.BBO.SBBOLOAD'//NOPADCHK +  
'SYS1.CDS.SGSKLOAD'//NOPADCHK)
```

Daemon and server control

Place on the access list of profile BPX.DAEMON in class FACILITY only daemons which are allowed to change their identity (eg. started tasks running OMVS, INETD, TCPIP, HTTP Server) :

```
PE BPX.DAEMON CL(FACILITY) ID(STUKERN STUINETD STUTCPIP STUWEBSV) ACC(READ)
```

Place on the access list of profile BPX.SERVER in class FACILITY only server IDs who are authorized to use the pthread_security_np() service (eg. started tasks running CICS Transaction Gateway and HTTP server):

PE BPX.SERVER CL(FACILITY) ID(STUCTG) ACC(READ)
PE BPX.SERVER CL(FACILITY) ID(STUWEBSV) ACC(UPDATE)

READ Means that both the server ID and the RACF ID of the client must be authorized to resources and the client must supply a password.

UPDATE Means that the server acts as surrogate of the client, that is, uses the identity and access granted to the client without the client's password being specified.

4.2.RACF groups for WebSphere

In this section we define all RACF groups for WebSphere.

WebSphere system programmers group

You may wish to appoint one or more MVS sysprogs or USS sysprogs (if you are lucky to have them) to install WebSphere with SMP/E and to run the ISPF customization dialogs, except the RACF job. Another solution is to assemble a brand new jobrole group from former MVS, CICS, IMS and DB2 sysprogs. The name for such group will depend on the shop's naming standard for jobrole group, but we will be using WASMNT.

AG WASMNT SUPGROUP(JOBROLE) OWNER(JOBROLE) OMVS(AUTOGID)
ALG WASMNT DATA('WAS SYSTEM PROGRAMMERS JOBROLE')
CO (userid1 userid2 etc) GROUP(WASMNT)

Note: You may already have selected a range of GIDs for your jobrole groups. If you have such range and wish to use AUTOGID then modify profile BPX.USER.NEXT in class FACILITY with your ranges for started task userids UIDs and jobrole GIDs as follows:

RDEF FACILITY BPX.NEXT.USER APPLDATA('your UID range st task/your GID range jobrole')

Irrespective of whether you use AUTOUID and AUTOGID or just assign the next available UID/GID in your range, you have to make sure that UID/GIDs are unique.

We found that their job functions require their group to be permitted to profile SUPERUSER.FILESYS.** in class UNIXPRIV with access CONTROL:

PE SUPERUSER.FILESYS. CL(UNIXPRIV) ID(WASMNT) ACC(CONTROL)**

Further in this article we will refer to this permit as Quasi Superuser (QSU).

WebSphere configuration group

The IBM WebSphere developers advise to create a group called configuration group consisting of all started task user IDs running the WebSphere procedures. This group must be the owning group permitted with rwx to all WebSphere runtime directories and files. Part of the design is to have a shared userid with a default name of WSADMIN which acts as Websphere administrator by logging to the Administrative console. This userid should be connected to this configuration group.

Define the configuration group with the command:

AG WASCFG SUPGROUP(STC) OWNER(STC) OMVS(AUTOGID)
ALG WASCFG DATA('WAS CONFIG JOBROLE FOR WAS ST TASK IDS')
CO (Daemon Controller Servants WSADMIN) GROUP(WASCFG)

Group WASCFG does not need to be permitted to any MVS resource. Also, because your WAS sysprogs (group WASMNT) are permitted to SUPERUSER.FILESYS.** in class UNIXPRIV, they do not need to be connected to group WASCFG.

Servants group

The command to create the jobrole group for servants started task user IDs is:

```
AG WASSRV SUPGROUP(STC) OWNER(STC) OMVS(AUTOGID)
ALG WASSRV DATA(' WAS JOBROLE FOR WAS SERVANTS ST TASK IDS')
```

Default Group for Machine User Ids used by USS

You may need to define a default group for machine (non-human user IDs used by Unix System Services(USS):

```
AG USSDFLT OMVS(AUTOGID) SUPGROUP(STC) OWNER(STC)
ALG USSDFLT DATA(' DEFAULT FOR MACHINE USER IDS USED BY USS')
```

WebSphere application administrators group

WebSphere application administrators play a powerful role in your organization. They will be deploying code, extracting logs, running scripts etc. We recommend appointing as WebSphere application administrators people with solid UNIX and MVS skills.

Define your jobrole group for WebSphere administrators with the command:

```
AG WASADM SUPGROUP(JOBROLE) OWNER(JOBROLE) OMVS(AUTOGID)
CO (userid1 userid2) GROUP(WASADM)
```

If you do not want to grant your WAS administrators QSU, then you must connect them to group WASCFG.

```
CO (members group WASADM) GROUP(WASCFG)
```

If your policy is not to have a mixture of human user IDs and machine user IDs connected to a jobrole group, then because you already made WASCFG the owning groups of all WebSphere HFSs, you need to create directory and file default ACLs for your mounting point, before generating the HFS:

Example 1 Access, directory and file default ACLs for group WASADM:

```
setfacl -m g:WASADM:rwx /WebSphere/cl01/AppServerNdA/
setfacl -m g:WASADM:rwx /WebSphere/cl01/DeploymentManager/
setfacl -m f:g: WASADM:rwx /WebSphere/cl01/AppServerNdA/
setfacl -m d:g:WASADM:rwx /WebSphere/cl01/DeploymentManager/
setfacl -m f:g: WASADM:rwx /WebSphere/cl01/AppServerNdA/
setfacl -m d:g:WASADM:rwx /WebSphere/cl01/DeploymentManager/
```

4.3.RACF user IDs for WebSphere

We will present commands defining started task user IDs for all procedures run in WebSphere Development/Test cells (our naming standard is to have T appended in the end of the userid for DEV/TEST and P - for PROD. Of course, if you have non-shared RACF database between DEV/TEST and PROD, you may scrap the T and P endings).

Daemon started task user ID

The command to create a user ID for any Location Service procedure (or Daemon) is:

```
AU STUWDD NAME('WAS DAEMON TEST') DFLTGRP(STC) OWNER(STC) NOPASSWORD +
OMVS(AUTOUID HOME('/u/stuwdd') PROG('/bin/sh'))
```

The Daemon user ID should be connected to group WASCFG:

CO STUWDD GROUP(WASCFG)

The Daemon user ID needs access to parameter libraries to retrieve CTRACE settings in the CTIBBO00 member, as follows:

PE 'SYS1.PARMLIB' GEN ID(STUWDD) ACC(R)

The Daemon user ID needs access to TCPIP parameter and load library:

PE 'SYS1.TCPPARMS' GEN ID(STUWDD) ACC(R)

PE 'SYS1.TCPIP.STANDARD.TCPXLBIN' GEN ID(STUWDD) ACC(R)

The Daemon user ID needs access to the following IBM programs (if they are covered by profile ** in class PROGRAM with UACC(READ) or id(*) ACC(READ), then you do nothing):

CEHUEY1	GSKSSL	IEEMB878	IOSTREAM
CEHUI1EY	GSKSUS31	IEFIB600	ITTCTSER
GSKCMS31	EDCZEMSG	IEFNB903	

If you use class SERVAUTH to protect TCPIP STACK and IP addresses, make sure that the Daemon user ID has access to profiles in class SERVAUTH of the type EZB.STACKACCESS and EZB.NETACCESS.

Controller started task user ID

The command to create the user ID that runs any Controller procedure is:

**AU STUWCD NAME('WAS CONTROLLER TEST') DFLTGRP(STC) OWNER(STC) NOPASS +
OMVS(AUTOUID HOME('/u/stuwcd') PROG('/bin/sh'))**

You may wish to create an alias and userid profile for the Controller user ID in case of data sets dumps:

DEFINE ALIAS (NAME('STUWCD') REL('user.catalog'))

AD 'STUWCD.'**

The Controller user ID should be connected to group WASCFG:

CO STUWCD GROUP(WASCFG)

The Controller user ID needs access to TCPIP parameter, load library and C/C++ compiler:

**PE 'SYS1.TCPPARMS' GEN ID(STUWCD) ACC(R)
PE 'SYS1.TCPIP.STANDARD.TCPXLBIN' GEN ID(STUWCD) ACC(R)
PE 'SYS1.TCPIP.HOSTS.SITEINFO' GEN ID(STUWCD) ACC(R)
PE 'SYS1.CBC.SCCNCMP' GEN ID(STUWCD) ACC(R)**

The Controller user ID needs access to the following IBM programs:

BPXBATC	GSKCMS31	IEFIB600	IEWBODEF
CEHUEY1	GSKSSL	IEFNB903	IGWASMS
CEHUI1EY	GSKSUS31	IOSTREAM	IGWASYS
EDCZEMSG	IEDCMSGT	ITTCTSER	

If you use class SERVAUTH to protect TCPIP STACK and IP addresses, make sure that the Controller user ID has access to profiles in class SERVAUTH of the type

EZB.STACKACCESS and EZB.NETACCESS.

Servant started task user ID

The command to create the userid that runs any Servant procedure is:

```
AU STUWSD NAME('WAS SERVANT TEST') DFLTGRP(STC) OWNER(STC) NOPASSWORD +  
OMVS(AUTOUID HOME('/u/stuwsd/') PROG('/bin/sh'))
```

The Servant user ID should be connected to the Servants group WASSRV and to WASCFG:

```
CO STUWSD GROUP(WASSRV)  
CO STUWSD GROUP(WASCFG)
```

The Servant user ID needs access to TCPIP parameter, load library and DB2 load libraries (if you plan to connect to DB2 from WebSphere)::

```
PE 'SYS1.TCPPARMS' GEN ID(STUWSD) ACC(R)  
PE 'SYS1.TCPIP.STANDARD.TCPXLBIN' GEN ID(STUWSD) ACC(R)  
PE 'SYS1.TCPIP.HOSTS.SITEINFO' GEN ID(STUWSD) ACC(R)  
PE 'DB2.subsystem.SDSNLOAD' GEN ID(STUWSD) ACC(R)
```

The Servant user ID needs access to the following IBM programs:

```
BPX1TAF      DSNHDECP    IEFIB600     IGWASYS  
CEHUEYI1     DSNRL1      IEFNB903     IGWASMS  
CEHUI1EY     DSN3ID00    IOSTREAM     ITTCTSER  
DSNARRS      EDCZEMSG    IEDCMSGT
```

If you use class SERVAUTH to protect TCPIP STACK and IP addresses, make sure that the Servant user ID has access to profiles in class SERVAUTH of the type EZB.STACKACCESS and EZB.NETACCESS.

CTRACE started task user ID

The command to create the user ID that runs the CTRACE procedure for any LPAR is:

```
AU STUWTR NAME('WAS TRACE WRITER') DFLTGRP(STC) OWNER(STC) NOPASSWORD
```

WAS Administrator user ID

This user ID (default name WSADMIN) has no password interval and should be defined with the NOEXPIRED password operand. The commands are:

```
AU WSADMIN NAME('WAS ADMINISTRATOR') DFLTGRP(USSDFLT) OWNER(USSDFLT) +  
OMVS(AUTOUID HOME('/tmp') PROG('/bin/sh'))  
ALU WSADMIN DATA('WAS ADMINISTRATOR SHARED')  
PW USER(WSADMIN) NOINTERVAL  
ALU WSADMIN PASSWORD(*****) NOEXPIRED
```

If you are going to share this user ID between the WebSphere administrators, it is recommended that you change its default password WSADMIN to a password of your choice.

This userid should be connected to group WASCFG:

```
CO WSADMIN GROUP(WASCFG)
```

Unauthenticated user ID

The unauthenticated user ID should be created with the RESTRICTED and NOPASSWORD attributes, because it will act on behalf of unauthenticated clients. The command to create the user ID is:

```
AU WASDFTU NAME('WAS UNAUTHENTICATED') DFLTGRP(USSDFLT) OWNER(USSDFLT) +  
OMVS(AUTOUID HOME('/tmp') PROG('/bin/sh')) RESTRICTED NOPASSWORD
```

Because this user has the RESTRICTED attribute, it must be explicitly permitted with READ to several resources having UACC(READ) or ID(*) with access READ. The most obvious are: profile ** in class PROGRAM, a few profiles starting with IRR.DIGTCERT in class FACILITY, and profile ** in class APPL (if you have it).

```
PE ** CL(APPL) ID(WASDFTU) ACC(READ)  
PE ** CL(PROGRAM) ID(WASDFTU) ACC(READ)
```

We recommend that profile ** in class APPL is deleted, if it has UACC>NONE, which is equivalent to not having this profile at all.

We recommend that you increase your default values in member BPXPRM00 in SYS1.PARMLIB for FILEPROCMAX to 10000, THREADSMAX to 10000, PROCUSERMAX to 5000.

The full list of keywords specifying various limits in the OMVS segment is shown in Table 2 below:

Table 2. Keywords and their values in OMVS segment

Parameter	Default value	Range	Unit
ASSIZEMAX	209,715,200	10M-2G	byte
CPUTIMEMAX	1000	7-2,147,483,647	sec
FILEPROCMAX	2,000	3-65535	number
MMAPAREAMAX	40,960	1-16,777,216	page
PROCUSERMAX	25	3-32,767	number
THREADSMAX	200	0-100000	number

4.4.Data set profiles for WebSphere

We recommend that all data sets needed for WebSphere be named with HLQ.WAS or with HLQ=WAS.

WebSphere data sets for ISPF dialogs

We defined one profile for our WebSphere MVS data sets and permitted our WebSphere system programmers group with ALTER, our WebSphere administrators with UPDATE, and our RACF Engineering group with READ:

```
AG WAS SUPGROUP(DATA) OWNER(DATA)  
DEF ALIAS (NAME('WAS') REL('user catalog'))  
AD 'WAS.**' OWNER(WASMNT)  
PE 'WAS.**' ID(WASMNT) ACC(ALTER)  
PE 'WAS.**' ID(WASADM) ACC(UPDATE)  
PE 'WAS.**' ID(RACFENG) ACC(READ)
```

This profile protects the CNFCFG, CNTL, and DATA data sets.

WebSphere HFS data sets

Our WebSphere HFS data sets OMVS.WAS.V500.CFG.*.HFS are protected by profile OMVS.WAS.**:

```
AD 'OMVS.WAS.**' OWNER (WASMNT)
```

```
PE 'OMVS.WAS.**' ID(WASMNT) ACC(ALTER)
PE 'OMVS.WAS.**' ID(STUKERN) ACC(UPDATE)
```

The last command is necessary if you decided that your OMVS proc should run without the TRUSTED flag.

WebSphere CTRACE data sets

To define a RACF profile for the WebSphere CTRACE data sets, issue the following command:

```
AD 'WAS.**.CTRACE.**' OWNER (WASMNT)
PE 'WAS.**.CTRACE.**' ID(WASMNT) ACC(ALTER)
PE 'WAS.**.CTRACE.**' ID(STUWTRC) ACC(UPDATE)
```

WebSphere data set for RACF commands

You may wish to create a data set in which to keep successive versions of member BBOWBRAK with RACF commands generated from the ISPF dialogs. The LSE RACF group can edit the commands according to our policies and then run them, in close cooperation with the WebSphere installers. To define a RACF profile for the WebSphere data sets containing RACF commands, issue the following command:

```
AD 'WAS.RACF.**' OWNER (RACFENG)
PE 'WAS.RACF.**' ID(RACFENG) ACC(ALTER)
PE 'WAS.RACF.**' ID(WASMNT) ACC(READ)
```

WebSphere log stream data sets

The names of the log stream data set name for WebSphere are of the type TWZ.WAS.ERROR.LOG.A0000xxx and TWZ.WAS.ERROR.LOG.A0000xxx.DATA, and to define a RACF profile for them, issue the following command:

```
AD 'TWZ.WAS.ERROR.LOG.**' OWNER(WASMNT)
PE 'TWZ.WAS.ERROR.LOG.**' ID(WASMNT) ACC(ALTER)
PE 'TWZ.WAS.ERROR.LOG.**' ID(STUWCD STUWSD) ACC(UPDATE)
PE 'TWZ.WAS.ERROR.LOG.**' ID(STUIXGL) ACC(UPDATE)
```

where STUIXGL is the started task user ID for your MVS IXGLOGR procedure and we assume that it is not running as TRUSTED.

RRS data sets

To define a RACF profile for the WebSphere RRS log stream data sets, issue the following commands:

```
AD 'WAS.RRS.**' OWNER (WASMNT)
PE 'WAS.RRS.**' ID(WASMNT) ACC(ALTER)
PE 'WAS.RRS.**' ID(STURRS) ACC(UPDATE)
```

We assume that you already have the RRS procedure running with STURRS user ID.

4.5.Profile in class JESSPOOL

In order to protect output from all started tasks and jobs related to WebSphere (they all start with WZ) we created a profile in class JESSPOOL :

```
RDEF JESSPOOL *.*WZ** OWNER(WASMNT)
PE *.*WZ** CL(JESSPOOL) ID(OPERGRP STUVPS WASADM) ACC(A)
PE *.*WZ** CL(JESSPOOL) ID(RACFENG) ACC(R)
```

4.6. Profiles in class STARTED

In this section we present the creation of profiles in class STARTED that match procedure member names in SYS1.PROCLIB for WebSphere daemons, controllers and servants. A profile is also created for the CTRACE procedure.

Profiles for daemon, controller, and servant procedures

To define profiles in class STARTED for procedures used by WebSphere with the TRACE option, issue the following commands:

For all Daemon procedures

```
RDEF STARTED WZD*.** OWNER(WASMNT) STDATA(USER(STUWDD) GROUP(STC))
```

For all Controller procedures

```
RDEF STARTED WZC*.** OWNER(WASMNT) STDATA(USER(STUWCD) GROUP(STC))
```

For the Servant procedures (Base server)

```
RDEF STARTED WZ%%%%%S.** OWNER(WASMNT) STDATA(USER(STUWSD) GROUP(STC))
```

For the Servant procedures (Deployment Manager)

```
RDEF STARTED WZ%%DMS.** OWNER(WASMNT) STDATA(USER(STUWSD) GROUP(STC))
```

Profile for CTRACE procedure

We created only one generic profile to cover all CTRACE procedures, issuing the command:

```
RDEF STARTED WZWTR*.** OWNER(WASMNT) STDATA(USER(STUWTRC) GROUP(STC))
```

4.7. Profiles in class APPL

WebSphere version 5 has a blanket application name CBS390 for all instances of run-time servers or clusters.

If you placed **Y** on Domain Security Identifier you have to create a matching profile in class APPL. For example you may wish to have different identifiers for your Unit testing and Integrated Testing environments. Create the following profiles (you may have one or more cells running with the same identifier):

```
RDEF APPL CBS390 OWNER(WASMNT)
```

```
RDEF APPL UNTEST OWNER(WASMNT)
```

```
RDEF APPL INTEST OWNER(WASMNT)
```

The access lists may look like this:

```
PE CBS390 CL(APPL) ID(WASCFG WASMNT WASADM sslapplid) ACC(READ)
```

4.8. Profiles in class LOGSTRM

IBM recommends that the log stream resource names match the installation's data set naming conventions, with userid or system logger application name as the first qualifier.

Usually the log stream resource name is made of the log stream data set name without its high level qualifier. The names of the log stream data set name for WebSphere are of the type TWZ.WAS.ERROR.LOG.A0000xxx and TWZ.WAS.ERROR.LOG.A0000xxx.DATA, and to define the corresponding log stream name in class LOGSTRM, we issued the command:

```
RDEF LOGSTRM WAS.** OWNER(WASMNT)
```

You have to authorize the group which creates the log streams (group WASMNT) with ALTER and the user IDs which write into the log streams (controller and servant user IDs) - with UPDATE. Group WASADM should have READ:

```
PE WAS.** CLASS(LOGSTRM) ID(WASMNT) ACCESS(ALTER)
PE WAS.** CLASS(LOGSTRM) ID(STUWCD STUWSD) ACCESS(UPDATE)
PE WAS.** CLASS(LOGSTRM) ID(WASADM) ACCESS(READ)
```

4.9.Profiles in class FACILITY

Protect LOGR policy

The LOGR policy information describes the characteristics of each log stream and coupling facility structure that will be used when there are active connections to the log stream. To authorize people who can create, change or delete LOGR policy, we defined profile MVSADMIN.LOGR in class FACILITY:

```
RDEF FACILITY MVSADMIN.LOGR OWNER(MVSMNT)
PE MVSADMIN.LOGR CL(FACILITY) ID(MVSMNT WASMNT) ACC(ALTER)
```

Authorize people who require reports on the LOGR policy with READ:

```
PE MVSADMIN.LOGR CL(FACILITY) ID(WASADM) ACC(READ)
```

For more information on log streams, coupling facility structures for sysplex, refer to ***Setting Up a Sysplex, SA22-7625***.

Protect Coupling Facility Structure

To protect the WebSphere coupling facility structure with name of WAS_SERV1 we issued the following commands:

```
RDEF FACILITY IXLSTR.WAS_SERV1 OWNER(WASMNT)
PERMIT IXLSTR.WAS_SERV1 CLASS(FACILITY) ID(WASMNT) ACCESS(ALTER)
```

Access to WLM policy

Workload management must be running in goal mode, and you must have access to a WLM definition, either saved in a WLM definition data set, or active in the WLM couple data set. The users of utility IWMARIN0 must have update access to profile MVSADMIN.WLM.POLICY in class FACILITY:

```
RDEF FACILITY MVSADMIN.WLM.** OWNER(MVSMNT)
PE MVSADMIN.WLM.** CL(FACILITY) ID(MVSMNT WASMNT) ACC(UPDATE)
```

FASTPATH processing

When profile BPX.SAFFASTPATH in class FACILITY is defined, RACF is not called if z/OS UNIX can quickly determine that file access will be successful. When the security product is bypassed, better performance is achieved, but successful file accesses cannot be audited. To define the profile issue the command:

```
RDEF FACILITY BPX.SAFFASTPATH UACC(NONE) OWNER(RACFENG)
```

Users do not need to be permitted to the BPX.SAFFASTPATH profile.

Access to Commands LIST Digital Certificates and LIST Key Rings

Several WebSphere started task IDs and administrators should be allowed to list their own Digital Certificates and Key Rings with the commands:

PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(STUWCD) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(STUWCD STUWSD) ACC(READ)

Note: Helpdesk can use RACDCERT ID(userid) LISTRING(*) to find all rings associated with a userid, but a similar command RACDCERT LIST(*) fails with a message

IKJ56712I INVALID KEYWORD, *
IKJ56703A REENTER THIS OPERAND - .

Access to IMS

IMS Open Transaction Manager Access (OTMA) is a transaction-based, connectionless client/server protocol. Though easily generalized, its implementation is specific to IMS in a sysplex environment. The domain of the protocol is restricted to the domain of the MVS Cross-System Coupling Facility (XCF). OTMA addresses the problem of connecting a client to a server so that the client can support a large network, or a large number of sessions, while maintaining high performance.

An MVS program in this case means any MVS application that is a member of an XCF group that includes IMS. The XCF group members that IMS communicates with are called OTMA clients. To protect XCF groups from any non-authorized caller create IMSXCF.OTMACI, a RACF resource, defined in the RACF facility class for the OTMA Callable Interface (C/I). When the RACF resource is defined, RACF RACHECK is invoked before OTMA C/I performs an XCF JOIN. This method protects the access to XCF, the XCF group, and the member.

This RACF checking is performed only when a non-authorized caller is using OTMA C/I. For more information refer to ***Open Transaction Manager Access Guide and Reference SC27-1303***.

To define the profile, issue the commands:

RDEF FACILITY IMSXCF.OTMACI OWNER(RACFENG)
PE IMSXCF.OTMACI CL(FACILITY) ID(WASSRV) ACC(READ)

4.10.Profiles in class CBIND

We can use the RACF class CBIND (optionally) to restrict the client's ability to access clusters. There are two types of profiles WebSphere for z/OS uses in the CBIND class:

- One that controls whether a local or remote client can "BIND" (access) a controller region in a cluster. The name of the profile has the format:
CB.BIND.cluster_name
where cluster_name is the name of the cluster.
- Another, that controls whether a client can use J2EE applications in a cluster. The name of the profile has the format:
CB.cluster_name
where cluster_name is the name of the cluster.

To define profiles in class CBIND for cluster names WZ0101 and WZ0201 (for Base cells) and WZ01DM and WZ02DM (for Deployment Managers) we created generic profiles:

RDEFINE CBIND CB.BIND.WZ* OWNER(WASMNT)
RDEFINE CBIND CB.WZ* OWNER(WASMNT)

We must authorize all WebSphere system management user IDs to have access READ to profiles CB.BIND.server_name and CB.cluster_name. The Controller user ID also needs access READ:

PERMIT CB.WZ* CLASS(CBIND) ID(WASADM WASMNT STUWCD) ACCESS(READ)
PERMIT CB.BIND.WZ* CLASS(CBIND) ID(WASADM WASMNT STUWCD) ACCESS(READ)

Note. A user ID can still gain access to the Controller Region if the session owner has access CONTROL. Within a local session (or SSL client certificate session) the session owner is the

user ID of the client or controller started task user ID that issued the message. Otherwise, ownership is assigned to the first user ID which has successfully accessed the controller region.

We have to grant access CONTROL to profile CB.WZ* in class CBIND if we need to indicate that a server is trusted to assert identities to WebSphere. Such servers should have already authenticated their callers. For example :

```
PERMIT CB.WZ* CLASS(CBIND) ID(STUWEBSV) ACCESS(CONTROL)
```

where SUWEBSV is the started task user ID for an IBM HTTP server.

If you placed Y on Domain Security Identifier you have to create matching profiles in class CBIND. For example you may wish to have different identifiers for your Unit testing and Integrated Testing environments. Create the following profiles for your Integrated Testing environment (you may have one or more cells running with the same identifier):

```
RDEFINE CBIND CB.INTEST.WZ* OWNER(WASMNT)  
RDEFINE CBIND CB.BIND.INTEST.WZ* OWNER(WASMNT)
```

4.11.Profiles in class SERVER

Servants must have access to profiles in the RACF SERVER class. This controls whether a servant can call authorized routines in the controller. Controllers do not require such access, because they run programs, loaded only from Authorized Program Facility (APF) libraries. The name of the profile has this form:

```
Format with 3 qualifiers: CB.<server_name>.<cluster_name>  
Format with 4 qualifiers: CB.<server_name>.<cluster_name>.<cell_name>
```

The 4-qualifier profile adds the cell name to avoid any ambiguities if more than one cell is defined.

We defined a generic profile to cover all possible server, cluster and cell names in the sysplex and placed the servants group with READ on its access list:

```
RDEF SERVER CB.*.WZ** OWNER(WASMNT)  
PERMIT CB.*.WZ** CLASS(SERVER) ID(WASSRV) ACC(READ)
```

4.12 Setting up SURROGAT check in CICS for remote program

If you plan to use the CICS Transaction Gateway Connector you may need to perform a surrogate user check to verify that the batch region 's userid is authorized to issue Distributed Program Link (DPL) calls for another user (that is, is authorized as a surrogate of the userid specified on the DPL_request call).

If you want your external CICS interface (EXCI) client jobs to be subject to surrogate user checking, specify SURROGCHK=YES in the EXCI options table, DFHXCOPT. If you specify SURROGCHK=YES, authorize the batch region 's userid as a surrogate of the userid specified on all DPL_request calls. For example, the following commands define a surrogate profile for a DPL userid, and grant READ access to the EXCI batch region:

```
RDEF SURROGAT dpl_userid.DFHEXCI OWNER(DPL_userid )  
PE userid.DFHEXCI CL(SURROGAT )ID(batch_region_userid) ACCESS(READ)
```

4.13.Profile in class OPERCMDS

The controller started task user ID must be able to issue MVS commands START and STOP against all servants procedures as well as its own, because these procedures are started internally from the controller procedures:

```
RDEF OPERCMDS MVS.**.WZ* OWNER(WASMNT)
PE MVS.**.WZ* CL(OPERCMDS) ID(STUWCD) ACC(UPDATE)
```

The above profile covers also the commands CANCEL, FORCE and MODIFY.

You may also consider to place your operator's group and WAS sysprog group on the access list of the above profile with access CONTROL (you would need this for command FORCE):

```
PE MVS.**.WZ* CL(OPERCMDS) ID(WASMNT OPERGRP) ACC(CONTROL)
```

Before creating the above profile, check that you do not have profiles of the type MVS.START.** , MVS.START.STC.** , MVS.STOP.** , MVS.STOP.STC.** etc, because they will take precedence over the above profile as more specific. We would recommend to delete them after transferring their access lists to MVS.**. If for some reason you cannot delete them, then you have to create five (5) specific profiles for each command of the type MVS.command.STC.WZ*.

The Controller and Servant user IDs may need to be able to route MVS commands to other LPARs:

```
PE MVS.ROUTE.CMD.** CL(OPERCMDS) ID(STUWCD STUWSD) ACC(READ)
```

4.14.Profiles in class EJBROLE

Profiles in RACF class EJBROLE (or grouping class GEJBROLE) are used to control a client's access to Enterprise Java Beans based on Role concept.

There are two distinct sets of tasks that are required to protect an application using EJB roles.

Note. Please, do not confuse the role concept (J2EE term) with the Job Role approach for building a RACF group tree structure.

The security administrator must define the roles as profiles in class EJBROLE or preferably in its grouping class GEJBROLE and permit selected groups/userids with access READ to them. There are four pre-defined roles for WebSphere Administration and we created respective profiles in class EJBROLE as follows:

```
RDEF EJBROLE administrator OWNER(WASMNT)
RDEF EJBROLE monitor OWNER(WASMNT)
RDEF EJBROLE configurator OWNER(WASMNT)
RDEF EJBROLE operator OWNER(WASMNT)
```

where the four roles above are pre-defined by IBM in low case. Roles utilized by applications can be defined in upper, lower or mixed case in the jar file. A role name cannot contain blanks, and cannot exceed 245 characters.

Note. If a security domain identifier was specified in the ISPF dialogs, eg INTEST, then you have to prefix the EJBROLE profiles with it – **INTEST.monitor** etc.

The four pre-defined roles are:

- Monitor** which can view configuration information and status but not anything more.
- Operator** which is a monitor that can trigger run time state changes, such as start an application server or stop an application, but cannot change configuration.
- Configurator** which is a monitor that can modify configuration information but cannot change run-time state.
- Administrator** which is an operator as well as a configurator.

The WebSphere system programmer or application developer must define all roles to be used by the application with name matching the RACF entities. Then the roles must be assigned **permission methods**.

Groups WASMNT and WASADM is permitted to all four roles with READ (the only meaningful access), while two other groups @WASMON and @WASCON are on the access list of roles monitor and configurator, respectively:

```
PERMIT administrator CLASS(EJBROLE) ID(WASADM STUWCD WASMNT) ACC(READ)
PERMIT monitor CLASS(EJBROLE) ID(WASADM WASMNT) ACC(READ)
PERMIT monitor CLASS(EJBROLE) ID(WASMON) ACC(READ)
PERMIT configurator CLASS(EJBROLE) ID(WASADM WASMNT) ACC(READ)
PERMIT configurator CLASS(EJBROLE) ID(WASCON) ACC(READ)
PERMIT operator CLASS(EJBROLE) ID(WASADM WASMNT) ACC(READ)
```

To protect CosNaming functions we created profiles for another set of four pre-defined by IBM roles in class EJBROLE:

```
RDEF EJBROLE CosNamingRead OWNER(WASMNT)
RDEF EJBROLE CosNamingWrite OWNER(WASMNT)
RDEF EJBROLE CosNamingCreate OWNER(WASMNT)
RDEF EJBROLE CosNamingDelete OWNER(WASMNT)
```

Only groups WASADM and WASMNT are permitted to all 4 roles with access READ which is the only meaningful access. User IDs WASDFTU and STUWSD also must be permitted to 2 profiles:

```
PERMIT CosNamingRead CL(EJBROLE) ID(WASDFTU WASADM WASMNT ) ACC(R)
PERMIT CosNamingWrite CL(EJBROLE) ID(WASADM WASMNT ) ACC(R)
PERMIT CosNamingCreate CL(EJBROLE) ID(WASADM WASMNT STUWSD) ACC(R)
PERMIT CosNamingDelete CL(EJBROLE) ID(WASADM WASMNT STUWSD) ACC(R)
```

Note. The creation of profile in class EJBROLE does not cause WebSphere to use them. That is controlled by the properties com.ibm.security.saf.authorization and com.ibm.security.saf.delegation, specified using the Administration console, on the custom properties page of the local OS user registry, under the security settings.

4.15.Profiles in class CSFSERV

The Controller ID STUWCD needs to be permitted to the following profiles in class CSFSERV related to SSL functions (for details on System SSL see <http://publibz.boulder.ibm.com/epubs/pdf/gska1a21.pdf>):

```
PE CSFCKI CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFCKM CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFENC CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFDEC CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFDSV CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFPKD CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFDSG CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFPKE CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFPKI CL(CSFSERV) ID(STUWCD) ACC(R)
```

4.16.RACF protection for connection of WebSphere to DB2

DB2 resources can be accessed by the WebSphere infrastructure and by Websphere applications. To add security to these resources, we can use the RACF DSNR resource class to protect DB2 resources. There are three functional areas in RACF to consider regarding protection for DB2:

Profile in class DSNR

RACF class DSNR controls access to the DB2 subsystems. Servants need access to the db2_ssn.RRSF profiles, where db2_ssn is the DB2 subsystem name or group sharing name.

RRSAF stands for Resource Recovery Service Attach Facility. If a servant does not have access, then its region will not initialize.

```
RDEF DSNR DTW1.RRSAF
PE DTW1.RRSAF CLASS(DSNR) ID(STUWSD) ACC(R)
```

Secondary authorization IDs

DB2 identification and signon exits (DSN3@ATH and DSN3@SGN) assign authorization IDs. If you want to use secondary authorization IDs (RACF groups), then you must replace the default exits with these two sample routines.

DB2 GRANT Statements

Unfortunately when connecting WebSphere for z/OS to DB2 the protection of DB2 objects through the DSNX@XAC exit (interface to RACF) is not recognized yet (V5.1). To protect DB2 objects, you must use GRANT statements.

Note. Recently a fix was provided to connect from WebSphere V5.1 to DB2 subsystems (V 7) converted to RACF security.

4.17. Setting permission for HFS files created by applications

Files created by applications running in the servant will have permission bits set according to the default umask. To change the default umask for the servant, specify the `_EDC_UMASK_DFLT` environment variable in the JCL procedure for the servant. On the JCL EXEC statement, specify:

```
PARM='ENVAR("_EDC_UMASK_DFLT=xxx")
```

where xxx is the umask value to use.

Recommendation: A umask value of 007 will cause files to be created with permission bits set to 770. This is the IBM recommended value.

5 Setting up SSL security for WebSphere for z/OS

5.1.SSL Objectives

The SSL support in WebSphere for z/OS has several objectives:

To provide ways accepted by the industry to protect the security of messages as they flow across the network. This is often called **transport layer security**. Transport layer security is a function that provides **privacy** and **data integrity** between two communicating applications. The protection occurs in a layer of software on top of the base transport protocol (for example, on top of TCP/IP). SSL provides security over the communications link through **encryption technology**, ensuring the **integrity** of messages in a network. Because communications are encrypted between two parties, a third party cannot tamper with messages. SSL also provides **confidentiality** (ensuring the message content cannot be read), **replay detection**, and **out-of-sequence detection**.

- To provide a secure communications medium through which various **authentication protocols** may operate. A single SSL session can carry multiple authentication protocols, that is, methods to prove the identities of the communicating parties. SSL support always provides a mechanism by which the server proves its identity. The SSL support on WebSphere for z/OS allows 4 ways for the client to prove its identity:
 - **Basic authentication (also known as SSL Type 1 authentication)**, in which a client proves its identity to the server by passing a **user identity and password** known by the target server. With SSL basic authentication:

1. A **z/OS client** can communicate securely with a WebSphere for z/OS server by using a user ID and password as defined by the CSiv2 Generic Security Services Username Password (GSSUP) mechanism.
 2. A **distributed platform client** can communicate securely with a WebSphere for z/OS server by using a **SAF user ID and password**.
 3. Because a password is always required on a request, only simple client-to-server connections can be made. That is, the **server cannot send a client's user ID to another server for a response to a request**.
 - **Client certificate support**, (also known as **mutual authentication**) in which both the server and client supply digital certificates to prove their identities to each other. Web applications may have thousands of clients, which makes managing client authentication an administrative burden. Through **RACF certificate name filtering**, SSL support on WebSphere for z/OS allows you to **map client certificates**, without storing them, to MVS user IDs. Through certificate name filtering, you can authorize sets of users to access servers without the administrative overhead of creating SAF user IDs and managing client certificates for every user.
 - **CSiv2 identity assertion support**, which includes z/OS principals, X501 distinguished names, Kerberos principals, and X509 identity certificates.
 - **Identity assertion, or trusted association**, in which an **intermediate server** can send the identities of its clients to a target server in a secure, yet efficient manner. This support uses client certificates to establish the intermediate server as the owner of an SSL session. Through SAF, the system can check that the intermediate server can be trusted (to confer this level of trust, RACF class CBIND authorization is granted by administrators to SAF IDs that run secure system code exclusively). Once trust in this intermediate server is established, client identities (SAF user IDs) need not be separately verified by the target server; those **client identities are simply asserted without requiring authentication**.
- **To interoperate in a secure way** with other products such as:
 - CICS Transaction server for z/OS
 - WebSphere on distributed platforms
 - CORBA-compliant Object Request Brokers

SSL is disabled by default and SSL support is optional. Running WebSphere for z/OS without using SSL affects only the SSL functions that protect communication and authenticate clients and servers. If you choose to use SSL, there are two types of SSL repertoires from which you must choose:

- **System SSL (SSSL)** is the SSL repertoire type used for Web container and ORB transport.
- **Java Secure Socket Extension (JSSE)** is the SSL repertoire type used for the JMX SOAP Connector.

The following describes how an SSL connection works:

Negotiation After the client locates the server, the client and server negotiate the type of security for communications. If SSL is to be used, the client is told to connect to a special SSL port.

Handshake The client connects to the SSL port and the SSL handshake occurs. If successful, encrypted communication starts. The client authenticates the server by inspecting the server's digital certificate. If client certificates are used during the handshake, the server authenticates the client by inspecting the client's digital certificate. Figure 9-1 on page 256 shows the flow and the components required to encipher and decipher data in an asymmetric handshake.

Ongoing communication During the SSL handshake, the client and server negotiate a cipher spec to be used to encrypt communications.

First client request The determination of client identity depends upon the client authentication mechanism chosen, which is one of the following:

- CSiv2 (Common Service Interoperability Version 2) user id and password (GSSUP)
- CSiv2 asserted identity
- zSAS (zOS Security Authentication Service) Kerberos
- z/SAS Basic Authentication Asserted Identities
- z/SAS Asserted Identities
- CSiv2 client certificates
- zSAS client certificates

Rules:

Only server controllers and z/OS clients require access to Cryptographic Services System SSL. Your controllers and z/OS clients require access to the *hlq.SGSKLOAD* data set. Place SGSKLOAD into LPA. For more information, see **System Secure Sockets Layer Programming, SC24-5901**.

- Either a Java or C++ client on z/OS can interoperate with a WebSphere for z/OS or workstation server and use SSL. CSiv2 security only supports Java clients on z/OS.
- Part of the handshake is to negotiate the cryptographic specs used by SSL for message protection. There are two factors that determine the cipher specs and key sizes used:
 - The security level of the Cryptographic Services installed on the system, which determines the cipher specs and key sizes available to WebSphere for z/OS
 - The configuration of the server through the Administrative Console allows you to specify SSL cipher suites.
- For z/OS System SSL sockets you must use RACF or equivalent for storing digital certificates and keys. Placing digital certificates and keys into a key database in the HFS is not an option.

The z/OS Security Server (RACF) RACDCERT command installs and maintains PKI private keys and certificates in RACF. Refer to the **z/OS: Security Server RACF Command Language Reference, SA22-7687** for details on the RACDCERT command. RACF supports multiple PKI private keys and certificates to be managed as a group. These groups are called **key rings**. RACF key rings are the preferred method for managing PKI private keys and certificates for System SSL.

5.2 SSL client certificate security for your WebSphere Application Server and clients

To define SSL client certificate security, you must first request **signed certificates for your server and clients, and certificate authority (CA) certificates** from the certificate authority that signed those certificates.

After you have received signed certificates and CA certificates from the certificate authority, you **must use RACF** (or another SAF-based registry) to authorize the use of digital certificates, store certificates and keyrings in RACF, and **define SSL security properties for your server through the Administrative Console. Each client identified by a digital certificate** must eventually be converted into a **SAF user ID** by the target WebSphere for z/OS server.

If the client and server share the same RACF database, then you do not have to do any additional configuration for this mapping. If the client and server do not share the same RACF database, you can configure the mapping by:

- Adding client certificates to the RACF database of the target server. This may be impractical in most cases.
- Mapping groups of clients into RACF identities using RACF certificate name filtering.
- Using a combination of the two.

The certificate arrangements involved in SSL client certificate authentication are:

- **For the client to authenticate the server**, the server (actually, the WAS controller user ID, which we named STUWCD) must possess a signed certificate created by a certificate authority (CA). The server passes the signed certificate to prove its identity to the client. The client must possess the CA certificate from the same certificate authority that issued the server's certificate. The client uses the CA certificate to verify that the server's certificate is authentic. Once verified, the client can be sure that messages are truly coming from that server, not someone else.
- **For the server to authenticate the client**, the client must possess a signed certificate created by a certificate authority (CA). The server must possess the CA certificate from the same certificate authority that issued the client's certificate. The server uses the CA certificate to verify that the client's certificate is authentic. Once verified, the server can be sure that messages are truly coming from that client, not someone else.

5.3 Defining SSL security for servers and clients

This section includes the procedures you must follow to implement all SSL-based authentication mechanisms.

Using RACF to authorize the server to use digital certificates

SSL uses digital certificates and public/private keys. If your WebSphere Application Server uses SSL, you **must use RACF to store digital certificates and public/private keys** for the user IDs under which the server controllers run. If you plan to implement SSL client certificate support, you must also have in RACF certificate authority (CA) certificates from each certificate authority that verifies your client certificates.

Perform the following steps authorizing the use of digital certificates by WAS on z/OS servers:

1. For each server that uses SSL, create a **keyring for that server's controller user ID**.
Tip: From z/OS 1.4 onwards it is possible to share both keyrings and certificate private keys. The keyrings may be accessed by other users for certificate checking purposes. Even the use of the private keys might be allowed to other user IDs.
 - To use someone else's keyring, specify the RACF USERID as the keyring prefix followed by a slash, e.g.: KEYFILE WEB001/my_keyring_name. The accessing USERID needs UPDATE access to IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING.
 - To share the same certificate (and private key) you need to:
 1. Generate (or add) the certificate under CERTAUTH or SITE.
 2. Connect the certificate to the keyring with USAGE(PERSONAL).
 3. Grant CONTROL access rights to IRR.DIGTCERT.GENCERT to the accessing server's USERID.
2. Receive a certificate for all server's controllers from a CA (for example, VeriSign). Store the certificate for each of your WAS cells in an MVS data set.
3. Connect the controller's certificate to the controller user ID's keyring and make the certificate the **DEFAULT certificate**.
4. If you plan to have the server authenticate clients (**SSL mutual authentication**), do the following:
 - Receive each CA certificate that verifies your client certificates. Give each CA certificate the CERTAUTH attribute.
 - Connect each client's CA certificate to the controller user ID's keyring.

By default, the following CAs are designated in RACF, with a status of **NOTRUST**.

- _ Integration Certification Authority Root
- _ IBM World Registry Certification Authority
- _ Thawte Personal Premium CA
- _ Thawte Personal Freemail CA
- _ Thawte Personal Basic CA
- _ Thawte Premium server CA
- _ Thawte server CA
- _ RSA Secure server Certification Authority

- _ Verisign Class 1 Public Primary Certification Authority
- _ Verisign Class 2 Public Primary Certification Authority
- _ Verisign Class 3 Public Primary Certification Authority

Before you can use any one of these CAs, you must first mark them with a status of **TRUST**, with the command:

```
RACDCERT CERTAUTH ALTER(LABEL('Verisign Class 3 Primary CA')) TRUST
```

Setting up SSL security for clients

All clients must have access to the controller's CA certificate, so they can authenticate the server during the SSL handshake. If you plan to implement SSL client certificate support, clients additionally must have their own certificates as the default certificate on their keyrings. If your clients are connecting to WebSphere for z/OS from WebSphere on workstations, you must import SSL certificates into the workstation system. For more information and instructions, see IBM WebSphere InfoCenter at:

http://publib.boulder.ibm.com/infocenter/wasinfo/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/twbs_secure.html

For SSL basic authentication and Kerberos, you must request a CA certificate from the **same certificate authority** that issued signed certificates for your controllers. If you plan to implement SSL client certificate support, you must additionally request a signed certificate for the client from a certificate authority.

RACF authorizations for SSL

Class APPL

If you specified "Security Domain Identifier" with a name of INTEST when you created the WebSphere cell, you had to create a profile in class APPL matching its name. You have to permit all your clients to this profile:

```
PE INTEST CL(APPL) ID(CLIENT1) ACC(READ)
```

If a userid is not authorized to to the above profile, the following message appears:

```
BBOS0056E initACEE (IRRSIA00) failed to convert SSL certificate with SAF
Return Code=8, RACF Return Code=8, RACF Reason Code=32.
```

```
BBOS0056E initACEE (IRRSIA00) failed to convert SSL certificate with SAF
Return Code=dstring, RACF Return Code=dstring, RACF Reason Code=dstring.
```

Explanation: The initACEE callable security service failed to convert an SSL Client Certificate with the indicated decimal return and reason codes.

User Response: See return codes for initACEE (IRRSIA00) in the Security Server (RACF) Callable Services or equivalent reference for other security products.

Class CBIND

If you specified "Security Domain Identifier" with a name of INTEST when you created the WebSphere cell, then, you have to authorize all your clients with access CONTROL to profile CB.BIND.INTEST.** in class CBIND, where ** represents all clusters in this cell:

```
PE CB.BIND.INTEST.** CL(CBIND) ID(CLIENT1) ACC(CONTROL)
```

You will want to ensure the certificates that will be accessing the server(s) via SSL have been mapped in RACF to a RACF userid. This is done by Certificate Name Filtering (CNF) or by loading the client certificate into RACF. The userid that is mapped to the client certificate will

then need access CONTROL to the following profile in class CBIND (you specified N on Security Domain Identifier):

```
PE CB.BIND.servername CLASS(CBIND) ID(clientCertUserid) ACCESS(CONTROL)
```

For example, to permit CLIENT1 access to server 01 in cell cl02 for a base app server:

```
PE CB.BIND.WZ0201 CLASS(CBIND) ID(CLIENT1) ACCESS(CONTROL)
```

Class EJBROLE

Many J2EE applications may require the creation of a role for mutual SSL authentication, eg. WaSSLUser. You have to create a profile in class EJBROLE and permit all your clients to it:

```
PE INTEST.WaSSLUser CL(EJBROLE) ID(CLIENT1) ACC(READ)
```

Note. You may have to use the corresponding "Security Domain Identifier" (in our case INTEST as a prefix to all profiles in class EJBROLE.

Mapping client digital certificates to MVS user IDs

Each client that presents a digital certificate to authenticate its identity, but does not have an individual certificate registered with RACF on the target server's system or cell, must have a mapping to a valid MVS user ID. You can create this mapping by using RACF certificate name filters. You can create RACF certificate name filters based on either the client's or certificate issuer's distinguished name, as contained in the X.509 digital certificates.

Perform the following steps to set up certificate name filtering:

1. Define an MVS user ID for each user ID you associate with a certificate name filter. Consider assigning the PROTECTED and RESTRICTED attributes to each one. The PROTECTED attribute protects the user ID from being used to logon directly to the system and from being revoked through incorrect password attempts. The RESTRICTED attribute ensures that the user ID will not be used to access resources with public access if he is not explicitly authorized.

```
AU WEBUSER DFLTGRP(MISC) OWNER(MISC) NOPASSWORD RESTRICTED
```

2. Create a certificate name filter. The following filter associates the user ID WEBUSER to any user presenting a certificate issued by VeriSign Class 1, who does not have an individual certificate registered with RACF on your system:

```
RACDCERT ID(WEBUSER) MAP WITHLABEL('INTERNET OTHERS')+  
IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign,Inc.L=Internet')
```

This filter is based on the issuer's name. You can create other filters based on the subject's name, or on combinations of the issuer's and subject's names. For more information about certificate name filtering, see *z/OS: SecurityServer RACF Security Administrators Guide, SA22-7683*.

5.4 Using certificates to set up HTTPS internal transport connections

An HTTPS internal transport can use server and client certificates to set up secure server-client connections for HTTPS application requests. The HTTPS internal transport enables you to set up client authentication using:

- Server certificates, created, signed and administered, by your own internal certificate authority (CA).

- Client certificates signed by an internal CA. Using an internal CA to sign your client certificates is independent of whether you used an internal or external CA to sign your server certificate.
- Server and cluster certificates signed by an external CA
- Client certificates that are signed by an external CA Using an external CA to sign your client certificates is independent of whether you used an internal or external CA to sign your server certificate.

Before you can use a server certificate to set up secure HTTPS internal transport connections, you must:

- Create or obtain a CA certificate, if you don't already have one.
- Create or obtain a server certificate, if you don't already have one.
- Create a server keyring to which you connect your server certificate, and has this certificate as the default for this keyring.
- Configure the HTTP transport from the administrative console.

If you also want to use a client certificate to set up secure HTTPS internal transport connections, you must perform the following additional tasks:

- Use the Administrative Console to specify that client certificates are allowed.
- Create or obtain a client certificate, if you don't already have one.

Setting up secure HTTPS internal transport connections using server certificates signed by an internal CA

Using SSL, WebSphere for z/OS allows you to set up your own CA and administer your own certificates:

- Acting as your own certificate authority (CA) is recommended only for **test environments and private intranets**. With this method, you set up your own CA and sign certificates with it. You can optionally use client authentication to verify the identity of those accessing your controller.
- You must use System SSL to establish secure connections.
- You will issue the RACF command, RACDCERT, to create certificates and keyrings for your J2EE server.
- Use the Administrators dialog in the Administrative Console to give an MVS ID administrative authority over a Java server. See **WebSphere Application Server for z/OS V5.1: System Administration, GA22-7962** for more information.

Perform these steps to set up secure connections using self-signed CA certificates:
In this section we will present the commands to create certificates and key rings for Base and Managed server controllers.

1. Create an internal (or self-signed) CA certificate:

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('WAS CertAuth for Security +
Domain') OU('WAS Sysprogs') O('antoff IT')) WITHLABEL('WebSphereCA') TRUST +
NOTBEFORE(DATE( 2005/08/21)) NOTAFTER(DATE(2010/08/21))
```

This CA certificate should be connected to the keyrings of all RACF user IDs having personal certificate. To list this certificate, we issue the following RACF command:

```
RACDCERT CERTAUTH LIST(LABEL('WebSphereCA'))
```

with the output:

Digital certificate information for CERTAUTH:

```
Label: WebSphereCA
Certificate ID: 2QiJmZmDhZmjgeaFguKXilWZhcPB
Status:TRUST
```

Start Date: 2004/01/21 00:00:00
 End Date: 2010/12/31 23:59:59
 Serial Number:
 >00<
 Issuer's Name:
 >CN=WAS CertAuth for Security Domain.OU=WAS Sysprogs<
 Subject's Name:
 >CN=WAS CertAuth for Security Domain.OU=WAS Sysprogs<
 Key Usage: CERTSIGN
 Private Key Type: Non-ICSF
 Private Key Size: 1024
 Ring Associations:
 Ring Owner: STUWCD
 Ring:
 >WASTKEYRING<

2. Export the CA certificate into an MVS dataset to make a backup:

```
RACDCERT CERTAUTH EXPORT(LABEL('WebSphereCA')) +
DSN('WAS.WASCA.CERT.BACKUP') FORMAT(PKCS12DER) PASSWORD('*****')
```

Important: Keyword of password must be specified only if the format of the certificate is PKCS#12.

3. You will need another dataset to store your CA public key for sending to clients:

```
RACDCERT CERTAUTH EXPORT(LABEL('WebSphereCA')) +
DSN('WAS.WASCA.CERT') FORMAT(CERTDER)
```

4. Create the WAS controller keyring:

```
RACDCERT ID(STUWCD) ADDRING(WASTKEYRING)
```

5. Connect your WAS CA certificate to the WAS controller keyring:

```
RACDCERT ID(STUWCD) CONNECT(RING(WASTKEYRING) +
LABEL('WebSphereCA') CERTAUTH)
```

6. Generate and sign your WAS controller certificate:

```
RACDCERT ID(STUWCD) GENCERT +
SUBJECTSDN(CN('WAS CONTROLLER') OU('INTEST') O('antoff IT')) +
SIGNWITH(CERTAUTH LABEL('WebSphereCA')) +
WITHLABEL('C.WZCL02') TRUST
```

Note. Our naming standard for OU is the main activity in the Cell, and for label – C.WZCLxx, where xx is the Cell number.

7. Connect your WAS controller certificate to the controller keyring:

```
RACDCERT ID(STUWCD) CONNECT +
(LABEL('C.WZCL02') RING(WASTKEYRING) DEFAULT)
```

8. Register the controller keyring with the Java server:

- a. Open the Administrative Console.
- b. Click **Servers > Application Servers** on the left-hand navigation tree.
- c. Click the name of the server.
- d. On the Additional Properties menu of the Server panel, click **Web Container**.
- e. On the Additional Properties menu of the Web Container panel, click **HTTP Transport**.
- f. Click the **Host** you want to configure.
- g. Enter the Port number you want to bind.

- h. Click the checkbox to **Enable SSL**.
 - i. Select the SSL alias in which you specified the controller keyring.
 - j. Click **OK**.
9. To verify that you can establish a secure connection with the controller, make sure the Java server is running, and then point your browser at the following URL:

`https://domain:port/dir/myapps/`

where:

domain is the domain where the Web application being requested resides.

port is the port number for SSL.

dir is the directory that contains the application.

myapps is the name of the certificate protected Web application being requested.

The first time you enter this URL, you should receive a warning that the CA certificate is not trusted. You will then be prompted to accept the certificate for the current request and all future requests. If you accept the certificate, it will be added to the browser's list of trusted CA certificates. The next time you enter this URL, you should not receive the warning.

- 10. Optionally, set up client authentication.

Setting up secure HTTPS internal transport connections using client certificates signed by an internal CA

Using SSL, WebSphere for z/OS allows you to set up client authentication using client certificates signed by an internal CA. You must ensure that the internal CA that signs your client certificates is marked with a status of TRUST and that it is connected to your controller keyring.

During the SSL handshake, the controller tells the client which CAs it trusts based on the trusted CAs in the controller keyring. The browser then searches its client certificates for ones issued by these CAs and allows the user to choose which client certificate to send to the controller.

Perform these steps to set up client authentication using client certificates signed by an internal CA (***we used these steps for our testing environments on the DEV sysplex***):

- 1. Create the client certificate and associate it with a RACF user ID.
Assuming your CA certificate was added to the list of trusted CAs in your browser, you can now generate a client certificate under a RACF user ID. This enables the client certificate to be used to authenticate the user ID.

```
RACDCERT ID(X-----) GENCERT +
SUBJECTSDN(CN('THEO ANTOFF') OU('antoff IT') O('RACF Consulting')) +
SIGNWITH(CERTAUTH LABEL('WebSphereCA')) +
WITHLABEL('CLIENT.TA') TRUST
```

The client certificate was created with status TRUST. Trust indicates that the client certificate can be used to authenticate the user ID X-----.

- 2. Create a keyring for RACF user ID X-----.

```
RACDCERT ADDRING(WASTKEYRING) ID(X-----)
```

- 3. Connect the client's certificate to the client's keyring.

```
RACDCERT ID(X-----) CONNECT +
```

(LABEL('CLIENT.TA') RING(WASTKEYRING) DEFAULT)

4. Connect the WAS CA certificate to the client's keyring:

```
RACDCERT ID(X-----) CONNECT +  
(LABEL('WebSphereCA') RING(WASTKEYRING) CERTAUTH)
```

5. Using the WebSphere for z/OS Administrative Console, verify that SSL client certificates are allowed:
 - a. Select **Conversations** > the name of the conversation > **Cell** > cellname > **Java Servers** > the name of the appropriate Java server > **Modify**.
 - b. In the properties form, check the **SSL Client Certificates** box, if it is not already checked, to indicate that SSL client certificates are allowed.
6. Ensure that the internal CA certificate is in the client's browser, if this is a requirement for the client's browser. (Some browsers do not require the CA certificate to reside in the browser.) The following example assumes it is not already there.

Follow the browser prompts to install the CA certificate. Newer versions of the Netscape and Microsoft® Internet Explorer browsers automatically start a wizard to help you install the certificate. If you use Microsoft Internet Explorer, you may need to open the file rather than saving it to disk to start the wizard. See the online help in your browser or browser documentation for additional information. Generally for Netscape, if you receive a window asking if you would like to accept the CA to certify network sites, electronic mail (e-mail) users, and software developers, do so.

7. Add the signed client certificate to the client's browser. To perform this step, you must:
 - a. Export the client certificate to an MVS data set.
Example: To export the client certificate to a data set so that the client certificate can be added to the client's browser, issue the following command:

```
RACDCERT ID(X-----) +  
EXPORT(LABEL('CLIENT.TA')) +  
DSN('X-----'.WAS.THEO.CERT') FORMAT(PKCS12DER) +  
PASSWORD('*****')
```

where:

X----- is the user RACF ID associated with the client certificate being exported.

'CLIENT.JB' is the label of the client certificate

'X-----'.WAS.THEO.CERT' is the data set that will contain the client certificate.

PKCS12DER indicates that the client certificate and private key are DER encoded when saved to the data set.

********* is the password associated with the encrypted client certificate. You will be required to provide this password when you import the client certificate into the browser. The password is case sensitive.

- b. FTP the client certificate to the client's workstation.

Example: This example shows how to use the FTP command to transfer the PKCS12 data set containing the signed client certificate to the client's workstation. The following steps are performed on the workstation:

- i. Enter the FTP command and the host name or IP address of the controller.
- ii. When prompted, enter your user ID (X-----) and password.
- iii. Enter **bin** to transfer the data set in binary format.
- iv. Transfer the data set to the workstation by entering: get 'X-----'.WAS.THEO.CERT' theocert.p12
- v. Enter quit or bye to exit.

An alternative to b. would be to use the IND\$FILE File Transfer Protocol to receive the dataset to your Windows directory using 'binary' and then email the cert with pfx extension to Theo's mail box.

c. Load the client certificate into the client's browser.

Example: This example shows how to load the PKCS12 file into the Microsoft Internet Explorer browser:

i. Start the browser.

ii. To access the security information, click **Tools>Internet Options>Content>Certificates**

iii. Under Certificates, click **Personal**.

iv. Click **Import a Certificate**.

v. Highlight the PKCS12 file.

vi. Click **Open**, and enter the case-sensitive password protecting the file.

vii. Click **OK**. The following message will be displayed: **Your certificates have been successfully imported.**

viii. Click **OK**. You should be able to see the following certificate label in the window: **These are your certificates**. You may need to scroll down to find the label.

8. Verify that the client can use the client certificate to access a protected page. To verify that the client can establish a secure connection with a protected page, make sure the Java server is running, and point your browser to the following URL
https://cl00t.xxx.xxx.com:xxxxx/webap1/my.jsp

When prompted by the browser, select the label for the client certificate. If the setup is correct, you will be able to view the protected page without prompts for a user ID and password.

Setting up secure HTTPS internal transport connections using server certificates signed by an external CA

Using SSL, WebSphere for z/OS allows you to use an external Commercial CA to sign your server certificate.

Perform these steps to set up secure connections using server certificates signed by an external CA (we will use a CA certificate sent by Valicert to ABC bank):

1. Ensure that your CA certificate is in the RACF list of CA certificates and marked with a status of TRUST. These conditions must be met before you can receive the CA-signed certificate into your controller keyring. In this step you must:
 - a. Check whether your external CA is in the list of CAs in RACF and whether it is marked with a status of TRUST.

RACDCERT CERTAUTH LIST

If your CA is in the list of CAs in RACF, but has a current status of NOTRUST, then change the status to TRUST. For example, issue the following command:

```
RACDCERT CERTAUTH ALTER(LABEL('ABC CORPORATION CERT AUTH')) TRUST
```

- b. If the CA cert of ABC bank is not in RACF, ask ABC to email you its public key. Save the ABC CA certificate to an MVS dataset with TRUST status:

```
RACDCERT CERTAUTH ADD('WAS.CERT.ABCCA') TRUST +  
WITHLABEL('ABC CORPORATION CERT AUTH')
```

2. Create the controller keyring:

```
RACDCERT ID(STUWCD) ADDRING(WASTKEYRING)
```

3. Generate the controller certificate. In this step you will:

- a. Create a self-signed certificate in order to establish the common name and public-private key pair.

```
RACDCERT ID(STUWCD) GENCERT +  
SUBJECTSDN(CN('WAS CONTROLLER') OU('INTTEST') O('ABC')) +  
WITHLABEL('C.WZCL01') TRUST
```

- b. Generate the controller certificate request from the self-signed certificate. You have to send this request to your external CA for signing by this CA.

```
RACDCERT ID(STUWCD) GENREQ(LABEL('C.WZCL01'))
```

- c. Export the certificate into an MVS dataset:

```
RACDCERT CERTAUTH EXPORT(LABEL('C.WZCL01')) +  
DSN('WAS.STUWCD.CL01.CERT') FORMAT(CERTDER)
```

4. Cut and paste the certificate request and email to your CA authority for signing.
5. Connect the external CA certificate to your controller keyring:

```
RACDCERT ID(STUWCD) CONNECT(RING(WASTKEYRING) +  
LABEL('ABC CORPORATION CERT AUTH') CERTAUTH)
```

6. Add the controller certificate signed by your external CA to RACF into the empty data set belonging to the self-signed certificate for Cell 01 and associate it with the STUWCD ID. In doing so, you will replace the self-signed certificate created in Step 3:

```
RACDCERT ID(STUWCD) ADD('WAS.STUWCD.CL01.CERT') WITHLABEL('C.WZCL01')
```

7. Connect your signed WAS controller certificate that is now in RACF to your WASTKEYRING keyring and make this certificate the default certificate in the keyring. Issue the following RACF command:

```
RACDCERT ID(STUWCD) CONNECT(ID(STUWCD) LABEL('C.WZCL01') +  
RING(WASTKEYRING) DEFAULT)
```

8. Register your controller keyring with the Application server.
 - a. Open the Administrative Console.
 - b. Click **Servers >Application Servers** on the left-hand navigation tree.
 - c. Click on the name of the **Server**.
 - d. On the **Additional Properties** menu of the **Server panel**, click **Web Container**.
 - e. On the **Additional Properties** menu of the **Web Container panel**, click **HTTP Transport**.
 - f. Click the **Host** you want to configure.
 - g. Enter the **Port** number you want to bind.
 - h. Click the checkbox to **Enable SSL**.
9. Verify that you can establish a secure connection with the controller. To do that, make sure the Java server is running, and point your browser at the same URL as in "Setting up secure HTTPS internal transport connections using client certificates signed by an internal CA".
10. Turn on client authentication. From the administration console, follow this path:
 - a. Security > SSL > alias-name (alias-name is normally **clnxx/DefaultHTTPS**)
 - b. On the configuration tab check the **Client Authentication** box.
 - c. On the configuration tab under **Cipher Suites** use the **Add >>** button to add all cipher suites.
 - d. Select the SSL alias in which you specified the controller keyring.
 - j. Click **OK**.
 - k. Under **Additional Properties** select **Custom Properties**.

- l. Click on **New**, then set:
 - m. Name to **MutualAuthCBindCheck**.
 - n. Value to **true**.
 - o. Description to **Require mutual authentication with client**
- p. Click **OK**.
- q. Update any other servers.
- r. Save to Master configuration.

Setting up secure HTTPS internal transport connections using client certificates signed by an external CA

WebSphere for z/OS allows you to set up client authentication using client certificates that are signed by an external CA. Using an external CA to sign your client certificates is independent of whether you used an internal or external CA to sign your server certificate.

You must ensure that the external CA that signs your client certificates is marked with a status of TRUST and that it is connected to your controller keyring. During the SSL handshake, the controller tells the client which CAs it trusts based on the trusted CAs in the controller keyring. The browser then searches its client certificates for ones issued by these CAs and allows the user to choose which client certificate to send to the controller.

Perform these steps to setup client authentication using client certificates signed by an external CA:

1. Create a keyring for each client:

RACDCERT ID(CLIENT1) ADDRING(WASTKEYRING)

2. Connect the CA certificate which signed the CLIENT1 certificate (let's say it has label ABC CORPORATION CERT AUTH) to the CLI1RING keyring owned by CLIENT1:

RACDCERT ID(CLIENT1) CONNECT(CERTAUTH + LABEL('ABC CORPORATION CERT AUTH') + RING(WASTKEYRING))

3. Verify that SSL client certificates are allowed.

a. Open the **Administrative Console**.

b. Click **Security > Authentication Protocols > CSlv2 Inbound Authentication** in the left-hand navigation tree.

c. In the General Properties section of the Configuration Tab for the CSlv2 Inbound Authentication panel, check to see if either Supported or Required is checked for Client Certificate Authentication.

4. Ensure that the CA certificate is in the client's browser. Browsers vary as to whether they require the CA certificate to be in the browser. This example assumes you will ensure that your CA certificate is added to your browser if it is not already there. If you are using the same external CA certificate to sign client certificates that you used to sign your server certificate, then clients may have already loaded the CA certificate into their browsers in step 5 of section, "Setting up secure HTTPS internal transport connections using server certificates signed by an external CA". If they have not, they should contact the external CA to obtain the CA certificate.
5. Obtain the client certificate. Follow the external CA's instructions for obtaining the signed client certificate and loading it into your browser.
6. Map a client certificate to a RACF user ID. RACF maps client certificates that are in various formats, including PKCS12, binary, and base-64 encoded ASCII. Some browsers may be able to output the client certificate in these formats or other formats.

If, for instance, either the **binary or base-64 encoded ASCII format is put out**, the resulting file would contain the client certificate **without the private key**. This is the format you need to map the client certificate to a RACF userid.

a. Ask the client to email you his certificate with the cer file extension (no private key). Use the IND\$FILE SEND command with 'text' option to obtain the certificate into an MVS data set WAS.CLIENT1.CERT.

b. Issue RACF commands to associate the client certificate with a RACF user ID exactly as in "Setting up SSL security for clients".

```
RACDCERT ID(CLIENT1) ADD('WAS.CLIENT1.CERT')+
WITHLABEL('CLIENT1.CERT')
```

7. Connect the certificate labelled CLIENT1.CERT to the keyring WASTKEYRING owned by CLIENT1, by issuing:

```
RACDCERT ID(CLIENT1) CONNECT(LABEL('CLIENT1.CERT')+
RING(WASTKEYRING) DEFAULT)
```

8. Verify that the client certificate is associated with a user ID that has been defined to RACF, by issuing the following RACDCERT command and specifying the user ID in the ID field:

```
RACDCERT ID(CLIENT1) LIST
```

9. To verify that a client can establish a secure connection with the controller, make sure the Java server is running, and point your browser at the same URL as in "Setting up SSL security for clients". When prompted by the browser, select the label for the client certificate. If the setup is correct, you will be able to view the protected page without prompts for a user ID and password.

You may get the following message if SSL was enabled, but a client or a server certificate are not found:

```
08.08.12 STC01356 BBOO0036E FUNCTION gsk_environment_init FAILED WITH RC=202.
```

6 CLIST to define RACF controls for WebSphere

Now, once you have got the whole picture you may wish to use the CLIST inbelow after careful examination of all RACF commands residing in it.

```
PROC 0
/*****/
/* RACF ENVIRONMENT FOR WebSphere */
/*
/* A sample CLIST to execute all RACF commands for installing
/* WebSphere
/* You must change this CLIST to satisfy your own shop standards
/* Comment out commands which you may decide not to run
/* Maintained by: your RACF/WAS administrators
/* History:
/* date
/*****/
/*
/* G R O U P S
/*
/*****/
/* AG STC SUPGROUP(DFLT) OWNNER(DFLT) OMVS(AUTOGID) */
/* ALG RACFENG OMVS(AUTOGID) */
/* ALG STC OMVS(AUTOGID) */
AG WASMNT SUPGROUP(JOBROLE) OWNER(JOBROLE) OMVS(AUTOGID)
CO (userid1 userid2 etc) GROUP(WASMNT)
AG WASADM SUPGROUP(JOBROLE) OWNER(JOBROLE) OMVS(AUTOGID)
CO (userid1 userid2 etc) GROUP(WASADM)
AG WASCFG SUPGROUP(STC) OWNER(STC) OMVS(AUTOGID)
AG WASSRV SUPGROUP(STC) OWNER(STC) OMVS(AUTOGID)
```

```

/*****/
/*
/* U S E R I D S
/*
/*****/
AU STUWDD NAME('WAS DAEMON') DFLTGRP(STC) OWNER(STC) +
NOPASSWORD OMVS(AUTOUID HOME('/u/stuwdd') PROG('/bin/sh'))
CONNECT (STUWDD) GROUP(WASCFG)
AU STUWCD NAME('WAS CONTROLLER') DFLTGRP(STC) OWNER(STC) +
NOPASSWORD OMVS(AUTOUID HOME('/u/stuwcd') PROG('/bin/sh'))
CONNECT (STUWCD) GROUP(WASCFG)
AU STUWSD NAME('WAS SERVANT') DFLTGRP(STC) OWNER(STC) +
NOPASSWORD OMVS(AUTOUID HOME('/u/stuwsd') PROG('/bin/sh'))
CONNECT (STUWSD) GROUP(WASSRV)
CONNECT (STUWSD) GROUP(WASCFG)
AU STUWRTC NAME('WAS TRACE WRITER') DFLTGRP(STC) OWNER(STC) +
NOPASSWORD
AU WSADMIN NAME('WAS ADMINISTRATOR') DFLTGRP(USSDFLT) OWNER(USSDFLT) +
OMVS(AUTOUID HOME('/u/wsadmin') PROG('/bin/sh'))
PW USER(WADMIN) NOINTERVAL
ALU WSADMIN PASSWORD(*****) NOEXPIRED
CONNECT WSADMIN GROUP(WASCFG)
AU WASDFTU NAME('WAS DEFAULT USER') DFLTGRP(USSDFLT) OWNER(USSDFLT) +
OMVS(AUTOUID HOME('/u/wasdfu') PROG('/bin/sh')) +
RESTRICTED NOPASSWORD
*****/
/*
/* z/OS UNIX
/*
/*****/
RDEF UNIXPRIV SUPERUSER.FILESYS.** OWNER(RACFENG)
PE SUPERUSER.FILESYS.** CL(UNIXPRIV) ID(WASMNT) ACC(CONTROL)
PE SUPERUSER.FILESYS.** CL(UNIXPRIV) ID(RACFENG) ACC(CONTROL)
RDEF UNIXPRIV SHARED.IDS UACC(NONE)
RDEF FACILITY BPX.SAFFASTPATH UACC(NONE) OWNER(RACFENG)
*****/
/*
/* D A T A S E T S
/*
/*****/
AG WAS SUPGROUP(DATA) OWNER(DATA)
DEF ALIAS (NAME('WAS') REL('user catalog name'))
AD 'WAS.**' OWNER(WASMNT)
PE 'WAS.**' ID(WASMNT) ACC(ALTER)
PE 'WAS.**' ID(WASADM) ACC(UPDATE)
PE 'WAS.**' ID(RACFENG) ACC(READ)
AD 'OMVS.WAS.**' OWNER (WASMNT)
PE 'OMVS.WAS.**' ID(WASMNT) ACC(ALTER)
AD 'OMVS.WAS.**' OWNER (WASMNT)
PE 'OMVS.WAS.**' ID(WASMNT) ACC(ALTER)
AD 'WAS.**.CTRACE.**' OWNER (WASMNT)
PE 'WAS.**.CTRACE.**' ID(WASMNT) ACC(ALTER)
PE 'WAS.**.CTRACE.**' ID(SUWTR) ACC(UPDATE)
AD 'WAS.RACF.**' OWNER (RACFENG)
PE 'WAS.RACF.**' ID(RACFENG) ACC(ALTER)
PE 'WAS.RACF.**' ID(WASMNT) ACC(READ)
AD 'TWZ.WAS.ERROR.LOG.**' OWNER(WASMNT)
PE 'TWZ.WAS.ERROR.LOG.**' ID(WASMNT) ACC(ALTER)
PE 'TWZ.WAS.ERROR.LOG.**' ID(STUWCD STUWSD) ACC(UPDATE)
PE 'TWZ.WAS.ERROR.LOG.**' ID(SUIXGL) ACC(UPDATE)
AD 'WAS.RRS.**' OWNER (WASMNT)
PE 'WAS.RRS.**' ID(WASMNT) ACC(ALTER)
PE 'WAS.RRS.**' ID(STURRS) ACC(UPDATE)
*****/
/*
/* CLASS JESSPOOL
/*
/*****/
RDEF JESSPOOL *.*WZ.** OWNER(WASMNT)
PE *.*WZ.** CL(JESSPOOL) ID(OPERGRP STUVPS WASADM) ACC(A)
PE *.*WZ.** CL(JESSPOOL) ID(RACFENG) ACC(R)
*****/
/*
/* CLASS STARTED
/*
/*****/
RDEF STARTED WZD.** OWNER(WASMNT) STDATA(USER(STUWDD) GROUP(STC))
RDEF STARTED WZC.** OWNER(WASMNT) STDATA(USER(STUWCD) GROUP(STC))
RDEF STARTED WZ%%S.** OWNER(WASMNT) STDATA(USER(STUWSD) GROUP(STC))
RDEF STARTED WZ%%DMS.** OWNER(WASMNT) STDATA(USER(STUWSD) GROUP(STC))
RDEF STARTED WZASH.** OWNER(WASMNT) STDATA(USER(STUADMSH) GROUP(STC))

```

```

RDEF STARTED WZVTR*.* OWNER(WASMNT) STDATA(USER(STUVTRC) GROUP(STC))
/*****
/*
/* CLASS APPL (OPTIONAL)
/*
/*****
RDEF APPL CBS390 OWNER(WASMNT)
RDEF APPL UNTEST OWNER(WASMNT)
RDEF APPL INTEST OWNER(WASMNT)
PE CBS390 CL(APPL) ID(WASCFG WASMNT WASADM sslapplid) ACC(READ)
/*****
/*
/* CLASS LOGSTRM
/*
/*****
RDEF LOGSTRM WAS.* OWNER(WASMNT)
PE WAS.* CLASS(LOGSTRM) ID(WASMNT) ACCESS(ALTER)
PE WAS.* CLASS(LOGSTRM) ID(STUWCD STUWSD) ACCESS(UPDATE)
PE WAS.* CLASS(LOGSTRM) ID(WASADM) ACCESS(READ)
/*****
/*
/* CLASS FACILITY (OPTIONAL)
/*
/*****
RDEF FACILITY MVSADMIN.LOGR OWNER(MVSMNT)
PE MVSADMIN.LOGR CL(FACILITY) ID(MVSSP WASMNT) ACC(ALTER)
PE MVSADMIN.LOGR CL(FACILITY) ID(WASADM) ACC(READ)
RDEF FACILITY IXLSTR.WAS_SERV1 OWNER(WASMNT)
PERMIT IXLSTR.WAS_SERV1 CLASS(FACILITY) ID(WASMNT) ACCESS(ALTER)
RDEF FACILITY MVSADMIN.WLM.* OWNER(MVSMNT)
PE MVSADMIN.WLM.* CL(FACILITY) ID(MVSMNT WASMNT) ACC(UPDATE)
RDEF FACILITY BPX.SAFFASTPATH UACC(NONE) OWNER(RACFENG)
RDEF FACILITY IRR.DIGTCERT.LIST OWNER(RACFENG)
RDEF FACILITY IRR.DIGTCERT.LISTRING OWNER(RACFENG)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(STUWCD) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(STUWCD STUWSD) ACC(READ)
/*****
/*
/* CLASS CBIND
/*
/*****
RDEFINE CBIND CB.WZ* OWNER(WASMNT)
RDEFINE CBIND CB.BIND.WZ* OWNER(WASMNT)
PERMIT CB.WZ* CLASS(CBIND) ID(WASADM) ACCESS(READ)
PERMIT CB.BIND.WZ* CLASS(CBIND) ID(WASADM) ACCESS(READ)
/*****
/*
/* CLASS SERVER
/*
/*****
RDEF SERVER CB.*.WZ*.* OWNER(WASMNT)
PERMIT CB.*.WZ*.* CLASS(SERVER) ID(WASSRV) ACC(READ)
/*****
/*
/* CLASS SURROGAT (OPTIONAL)
/*
/*****
RDEF SURROGAT dpl_userid.DFHXC1 UACC(NONE) OWNER(dpl_userid)
PE userid.DFHXC1 CL(SURROGAT) ID(batch_region_userid) ACCESS(READ)
/*****
/*
/* CLASS OPERCMDS
/*
/*****
RDEF OPERCMDS MVS.*.WZ* OWNER(WASMNT)
PE MVS.*.WZ* CL(OPERCMDS) ID(STUWCD) ACC(UPDATE)
PE MVS.*.WZ* CL(OPERCMDS) ID(WASMNT OPERGRP) ACC(CONTROL)
/*****
/*
/* CLASS EJBROLE
/*
/*****
RDEF EJBROLE administrator OWNER(WASMNT)
RDEF EJBROLE monitor OWNER(WASMNT)
RDEF EJBROLE configurator OWNER(WASMNT)
RDEF EJBROLE operator OWNER(WASMNT)
PE administrator CLASS(EJBROLE) ID(WASADM WASMNT STUWCD) ACC(READ)
PE monitor CLASS(EJBROLE) ID(WASADM WASMNT) ACC(READ)
PE monitor CLASS(EJBROLE) ID(WASMON) ACC(READ)
PE configurator CLASS(EJBROLE) ID(WASADM WASMNT) ACC(READ)
PE configurator CLASS(EJBROLE) ID(WASCON) ACC(READ)

```

```

PE operator CLASS(EJBRLE) ID(WASADM WASMNT) ACC(READ)
RDEF EJBRLE CosNamingRead OWNER(WASMNT)
RDEF EJBRLE CosNamingWrite OWNER(WASMNT)
RDEF EJBRLE CosNamingCreate OWNER(WASMNT)
RDEF EJBRLE CosNamingDelete OWNER(WASMNT)
PE CosNamingRead CL(EJBRLE) ID(WASDFTU WASADM WASMNT STUWSD) ACC(R)
PE CosNamingWrite CL(EJBRLE) ID(WASADM WASMNT STUWSD) ACC(R)
PE CosNamingCreate CL(EJBRLE) ID(WASADM WASMNT) ACC(R)
PE CosNamingDelete CL(EJBRLE) ID(WASADM WASMNT) ACC(R)
/*****
/*
/* CLASS CSFSERV
/*
/*****
PE CSFCKI CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFCKM CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFENC CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFDEC CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFDSV CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFPKD CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFDSG CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFPKE CL(CSFSERV) ID(STUWCD) ACC(R)
PE CSFPKI CL(CSFSERV) ID(STUWCD) ACC(R)
/*****
/*
/* CLASS DSNR
/*
/*****
RDEF DSNR DTW1.RRSF
PE DTW1.RRSF CLASS(DSNR) ID(STUWCD STUWSD) ACC(R)
/*****
/*
/* CERTIFICATE ADMINISTRATION
/*
/*****
AG HELPDESK SUPGROUP(JOBROLE) OWNER(JOBROLE) OMVS(AUTOGID)
AU HDUSR1 DFLTGRP(EMPL) OWNER(EMPL) OMVS(AUTOUID)
CO HDUSR1 GROUP(HELPDESK)
PE IRR.DIGTCERT.LIST CL(FACILITY) ID(HELPDESK) ACC(C)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(HELPDESK) ACC(C)
/*****
/*
/* SSL FOR SERVERS (WAS CONTROLLERS) WITH INTERNAL CA
/*
/*****
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('WAS CertAuth for Security +
Domain') OU('WAS Sysprogs') O('antoff IT'))WITHLABEL('WebSphereCA') +
TRUST NOTBEFORE(DATE( 2005/08/21)) NOTAFTER(DATE(2010/08/21))
RACDCERT CERTAUTH EXPORT(LABEL('WebSphereCA')) +
DSN('WAS.WASCA.CERT.BACKUP') FORMAT(PKCS12DER) PASSWORD('*****')
RACDCERT CERTAUTH EXPORT(LABEL('WebSphereCA')) +
DSN('WAS.WASCA.CERT') FORMAT(CERTDER)
RACDCERT ID(STUWCD) GENCERT +
SUBJECTSDN(CN('WAS CONTROLLER') OU('INTEST') O('antoff IT')) +
SIGNWITH(CERTAUTH LABEL('WebSphereCA')) +
WITHLABEL('C.WZCL02') TRUST
RACDCERT ADDRING(WASTKEYRING) ID(STUWCD)
RACDCERT ID(STUWCD) CONNECT +
(LABEL('C.WZCL02') RING(WASTKEYRING) DEFAULT)
RACDCERT ID(STUWCD) CONNECT(RING(WASTKEYRING) LABEL('WebSphereCA') +
CERTAUTH)
/*****
/*
/* SSL FOR CLIENTS WITH INTERNAL CA
/*
/*****
RACDCERT ID(X-----) GENCERT +
SUBJECTSDN(CN('THEO ANTOFF') OU('antoff IT') O('RACF Consulting')) +
SIGNWITH(CERTAUTH LABEL('WebSphereCA')) +
WITHLABEL('CLIENT.TA') TRUST
RACDCERT ADDRING(WASTKEYRING) ID(X-----)
RACDCERT ID(X-----) CONNECT +
(LABEL('CLIENT.TA') RING(WASTKEYRING) DEFAULT)
RACDCERT ID(X-----) CONNECT +
(LABEL('WebSphereCA') RING(WASTKEYRING) CERTAUTH)
RACDCERT ID(X-----) +
EXPORT(LABEL('CLIENT.TA')) +
DSN('X-----.WAS.THEO.CERT') FORMAT(PKCS12DER) +
PASSWORD('*****')
/*****
/*

```

```

/* SSL FOR SERVERS WITH EXTERNAL CA */
/* */
/*****/
RACDCERT CERTAUTH ALTER(LABEL('ABC CORPORATION CERT AUTH')) TRUST
or
RACDCERT CERTAUTH ADD('WAS.CERT.ABCCA') TRUST +
WITHLABEL('ABC CORPORATION CERT AUTH')
RACDCERT ID(STUWCD) ADDRING(WASTKEYRING)
RACDCERT ID(STUWCD) GENCERT +
SUBJECTSDN(CN('WAS CONTROLLER') OU('INTTEST') O('ABC')) +
WITHLABEL('C.WZCL01') TRUST
RACDCERT ID(STUWCD) GENREQ(LABEL('C.WZCL01'))
RACDCERT CERTAUTH EXPORT(LABEL('C.WZCL01')) +
DSN('WAS.STUWCD.CL01.CERT') FORMAT(CERTDER)
RACDCERT ID(STUWCD) CONNECT(RING(WASTKEYRING) +
LABEL('ABC CORPORATION CERT AUTH') CERTAUTH)
RACDCERT ID(STUWCD) ADD('WAS.STUWCD.CL01.CERT') WITHLABEL('C.WZCL01')
RACDCERT ID(STUWCD) CONNECT(ID(STUWCD) LABEL('C.WZCL01') +
RING(WASTKEYRING) DEFAULT)
/*****/
/* */
/* SSL FOR CLIENTS WITH EXTERNAL CA */
/* */
/*****/
RACDCERT ID(CLIENT1) ADDRING(WASTKEYRING)
RACDCERT ID(CLIENT1) CONNECT(CERTAUTH +
LABEL('ABC CORPORATION CERT AUTH')+
RING(WASTKEYRING))
RACDCERT ID(CLIENT1) ADD('WAS.CLIENT1.CERT') +
WITHLABEL('CLIENT1.CERT') TRUST
RACDCERT ID(CLIENT1) CONNECT(LABEL('CLIENT1.CERT') +
RING(WASTKEYRING) DEFAULT)
/*****/
/* */
/* SETR REFRESH */
/* */
/*****/
SETR GENERIC(DATASET) REFR
SETR RACLIST(CBIND) REFR
SETR RACLIST(SERVER) REFR
SETR RACLIST(CSFSESV) REFR
SETR RACLIST(CSFKEYS) REFR
SETR RACLIST(EJBRLE) REFR
SETR RACLIST(FACILITY) REFR
SETR RACLIST(STARTED) REFR
SETR RACLIST(SURROGAT) REFR
SETR WHEN(PROGRAM) REFR
/*****/

```