

IBM Operational Decision Manager
Version 8 Release 6

*Configuring Operational Decision
Manager on z/OS*



Note

Before using this information and the product it supports, read the information in "Notices" on page 139.

This edition applies to version 8, release 6, modification 0 of Operational Decision Manager and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2014, 2014.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Configuring 1

Step 1: Authorizing the load library. 1

Step 2: Choosing your topology 2

Topology 1: zRule Execution Server group with a shared console. 2

Topology 2: one or many CICS rule-owning regions 3

Topology 3: CICS rule-owning and application-owning regions 4

Topology 4: Operational Decision Manager on WebSphere Application Server for z/OS 6

Topology 5: batch execution and an embedded zRule Execution Server for z/OS instance 8

Step 3: Customizing the z/OS configuration and runtime variables. 9

Step 4: Customizing the control statements for your topology. 9

Customizing topology 1: zRule Execution Server group with a shared console. 10

Customizing topology 2: one or many CICS rule-owning regions 11

Customizing topology 3: CICS rule-owning and application-owning regions 12

Customizing topology 4: Operational Decision Manager on WebSphere Application Server for z/OS 13

Customizing topology 5: batch execution and an embedded zRule Execution Server for z/OS instance 13

Step 5: Creating the working data sets 14

Step 6: Creating the working directory 15

Step 7: Configuring a DB2 persistence layer 15

Creating a DB2 database for your rules runtime persistence layer. 15

Creating the event runtime database for z/OS. 16

Creating the Decision Center DB2 database. 17

Step 8: Configuring your topology. 18

Configuring topology 1: zRule Execution Server group with a shared console. 18

Configuring a zRule Execution Server for z/OS server group 18

Configuring CICS to execute rules on zRule Execution Server for z/OS 19

Configuring IMS to execute rules on zRule Execution Server for z/OS 20

Securing zRule Execution Server for z/OS resources 22

Security options 22

Securing access to the working directory and the installation directory 22

Managing server security. 23

Starting a new server instance 29

Configuring topologies 2 and 3: CICS rule and application-owning regions 29

Configuring a CICS rule-owning region to execute rules in a CICS JVM server 29

Configuring a CICS application-owning region to execute rules in a CICS rule-owning region . 32

Configuring topology 4: Operational Decision Manager on WebSphere Application Server for z/OS 33

Configuring security on WebSphere Application Server for z/OS. 34

Configuring RACF security 34

Configuring a federated repository 37

Configuring Rule Execution Server on WebSphere Application Server for z/OS. 41

Automated configuration of Rule Execution Server on WebSphere Application Server for z/OS 41

Manual configuration of Rule Execution Server on WebSphere Application Server for z/OS 43

Optional configuration steps. 53

Configuring Decision Server Events on WebSphere Application Server for z/OS. 70

Stand-alone configuration on WebSphere Application Server for z/OS. 71

Configuring the event runtime to run on WebSphere Application Server 73

Using Event Designer with z/OS 74

Verifying your Events z/OS configuration 74

Tuning your z/OS configuration 78

Configuring additional options for Decision Server Events for z/OS 80

Configuring additional options for the File System Connector 80

Uninstalling on z/OS 81

Configuring the Decision Center consoles on WebSphere Application Server for z/OS. 81

Automated configuration of Decision Center on WebSphere Application Server for z/OS 81

Manual configuration of Decision Center on WebSphere Application Server for z/OS. 83

Verifying your configuration of Decision Center 96

Configuring topology 5: batch execution and an embedded zRule Execution Server for z/OS instance 98

Configuring the Java batch environment to run a Java batch job 98

Configuring the Java batch environment. 98

Running the sample Java batch program. 99

Step 9: Verifying your rules execution environment configuration 99

Configuring a COBOL rule subprogram into a COBOL application 100

COBOL-generated code integration 100

Linking code by using a COBOL static link 100

Linking code by using a COBOL dynamic link 102

Calling a CICS wrapper program to integrate the code	103	Execution data that is collected in the SMF records	111
Configuring your system to collect usage data	104	Configuring zRule Execution Server for z/OS to collect execution data.	113
Configuring zRule Execution Server for z/OS to collect usage data	104	Configuring a CICS rule-owning region to collect execution data.	113
Configuring Rule Execution Server on WebSphere Application Server for z/OS to collect data	104	Configuring Rule Execution Server on WebSphere Application Server for z/OS to collect execution data.	114
Configuring COBOL rule subprograms to collect data	105	Configuring Java batch to collect execution data	115
Usage monitoring tools	105	Sample code to format execution data	115
Configuring z/OS.	106	Sample code to export execution data in CSV format.	116
Configuration tasks	106	z/OS configuration variables	117
Defining the Resource Manager as a z/OS subsystem	107	z/OS runtime variables	126
Loading the COBOL load module into the LPA	107	z/OS data sets and directories.	136
Configuring the Resource Manager subsystem	108	HBRUUPI job.	137
Compiling, linking, and running your programs.	109	Notices	139
Sample COBOL code for usage data collection	110	Trademarks	141
Configuring your system to collect execution data	111	Index	143
Collection of execution data by using SMF type 120 subtype 100 records	111		

Configuring

You must configure Operational Decision Manager for z/OS® before you can send requests to the rule engine from your applications. If your execution environment is on WebSphere® Application Server for z/OS you must deploy and configure the provided archives for Rule Execution Server and the event runtime. You can also deploy the Decision Center consoles on WebSphere Application Server for z/OS to allow users to author and test rule projects.

To set up a Decision Server execution environment or Decision Center, you must customize the data sets, z/OS configuration variables, and z/OS jobs provided, and then use them to configure new environments. Depending on your requirement, you could use identical environments to maximize execution throughput or you could use one type of environment for legacy rules and another type for rules authored in Decision Center. The configuration process consists of initial steps that are common to all execution environments, followed by steps that are specific to your environment.

Decision Server supports the following execution environments:

- zRule Execution Server group with a shared console
- One or many CICS® rule-owning regions
- CICS rule-owning and application-owning regions
- Operational Decision Manager on WebSphere Application Server for z/OS
- Batch execution and an embedded zRule Execution Server for z/OS instance

Decision Center provides a single repository, which can be deployed to production mainframe applications. Decision Center for z/OS works in the same way as Decision Center does, except that it runs on WebSphere Application Server for z/OS.

Step 1: Authorizing the load library

After you install the Decision Server for z/OS data sets, you must make the ++HBRHLQ++.SHBRAUTH load library APF-authorized.

About this task

Important: If Decision Server Events is the only Decision Server for z/OS execution environment you plan to use is, omit this step.

If the load library is not already authorized, add the Decision Server for z/OS load library to the list of authorized libraries.

If you execute your rules in a CICS or IMS™ environment by using WebSphere Optimized Local Adapters (WOLA), the WOLA load libraries must also be APF-authorized. For more information, see “Configuring CICS to run rules through WOLA” on page 56 and “Configuring IMS to execute rules through WOLA” on page 58.

Procedure

1. Add the ++HBRHLQ++.SHBRAUTH load library to the list of APF-authorized libraries by using the following command:

```
SETPROG APF,ADD,DSNAME=++HBRHLQ++.SHBRAUTH,SMS
```

where ++HBRHLQ++ represents the high-level qualifier that is assigned to Decision Server data sets during installation.

Consider adding a definition to the system parameter library (parmlib) member PROGxx to make it permanent. For example:

```
APF ADD DSNAME(++HBRHLQ++.SHBRAUTH) SMS
```

2. Provide RACF® authorization for the ++HBRHLQ++.SHBRAUTH load library by using the following commands:

```
RALTER PROGRAM * ADDMEM('++HBRHLQ++.SHBRAUTH'//NOPADCHK)
```

```
SETOPTS WHEN(PROGRAM) REFRESH
```

Step 2: Choosing your topology

You can configure Operational Decision Manager for z/OS in different topologies that are based on your production needs and your existing configurations.

Topology 1: zRule Execution Server group with a shared console

This topology consists of a server group that contains multiple instances. The topology also contains a shared Rule Execution Server console. Each component runs in a separate address space. A DB2® database is used as a persistence layer that stores rule applications and data.

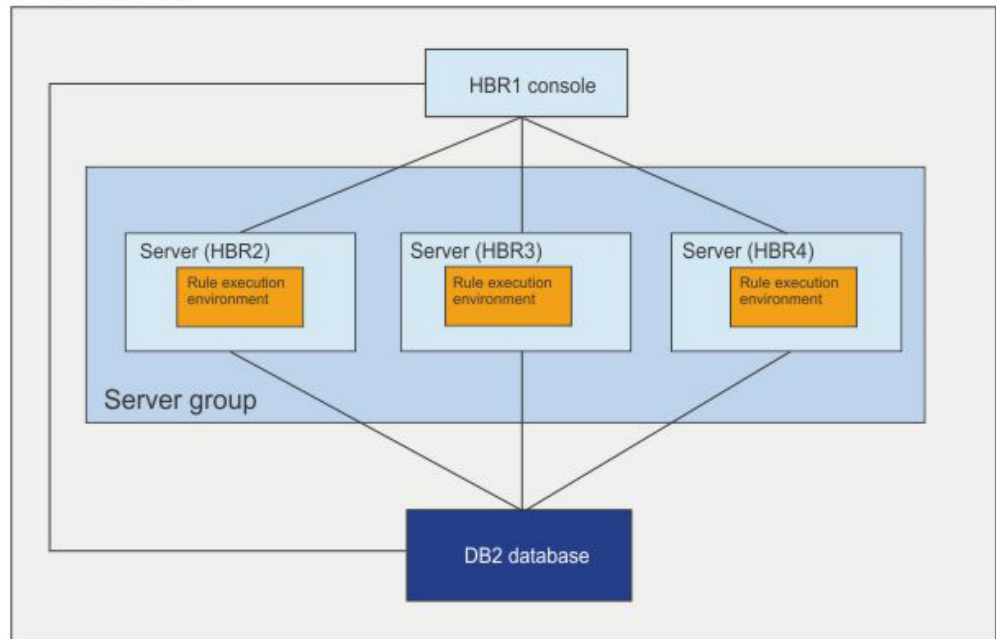
Reasons for choosing this topology

- To call rules from batch, IMS, or CICS applications in a zRule Execution Server for z/OS that runs in the same LPAR.
- If a server error occurs, for rules execution failover to another zRule Execution Server for z/OS in the group.
- To support batch jobs and IMS jobs that call rule applications (CICS applications can run against the zRule Execution Server for z/OS instead of using a CICS JVM server).

Architecture

The example shows a group of three servers with a single console. A group can contain a maximum of 32 servers. The servers connect to the same database and console.

z/OS LPAR1



If a server in the group becomes inactive, rule execution transfers to the next available server in the group. The inactive server remains a member of the server group. When the inactive server restarts, rule execution transfers back to it. These transfers of rule execution processing do not require any action by the z/OS, COBOL or PL/I application.

A zRule Execution Server for z/OS instance on a CICS JVM server consists of a server group that contains a single server. In this situation, if the server becomes inactive, rule execution transfer to another server is not possible.

The following processes occur when z/OS applications run rules in a server group:

1. Batch jobs and IMS applications use the HBRCONN API to connect to the servers specified in the list. CICS applications use the HBRC transaction to connect to the servers specified in the list. All servers in the group must be capable of executing rules, all servers in the server list must be running, and the connection method must return a completion code of HBR_CC_0K (0).
2. z/OS applications use a single HBRRULE API call to run each ruleset.
3. When the rulesets have completed, each z/OS application uses an HBRDISC API call to disconnect from its Rule Execution Server. The **HBRA-CONN-SSID** field in the HBR-CONN-AREA data area returns the subsystem ID of the server that completed the execution of the ruleset. If the rulesets did not run to completion, the field is set to blanks.

Topology 2: one or many CICS rule-owning regions

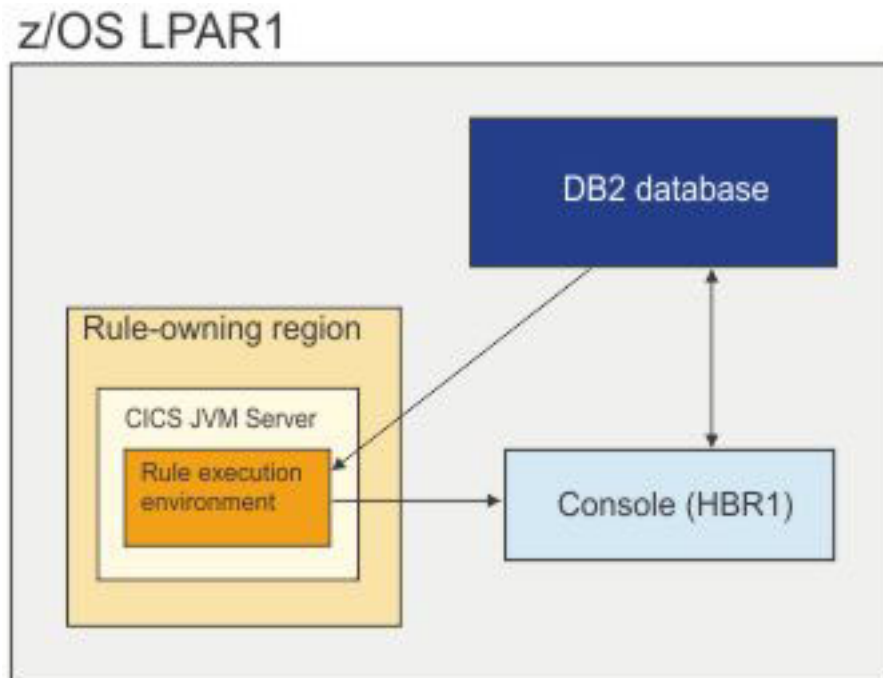
This topology includes a Rule Execution Server console that runs in its own address space. The execution units (XU) are deployed to one or more CICS JVM servers in a rule-owning region. A DB2 database is used as a persistence layer to store rule applications and data.

Reasons for choosing this topology

- To run rules in multiple regions, and to run a rules workload that justifies the memory or CPU requirements of running a CICS JVM server in every region.
- To run rules inside a CICS JVM server.
- To enable regions on CICS Transaction Server V4.2 or later to run rules.
- To enable CICS applications to connect to a zRule Execution Server for z/OS instance to run rules.
- To ensure the high availability of rules by using CICSplex® workload manager (WLM).

Architecture

If more rule-owning regions are added to this architecture, they connect to the same DB2 database and Rule Execution Server console.



Topology 3: CICS rule-owning and application-owning regions

This topology uses two different types of CICS region (the rule-owning region and the application-owning region) to run rules.

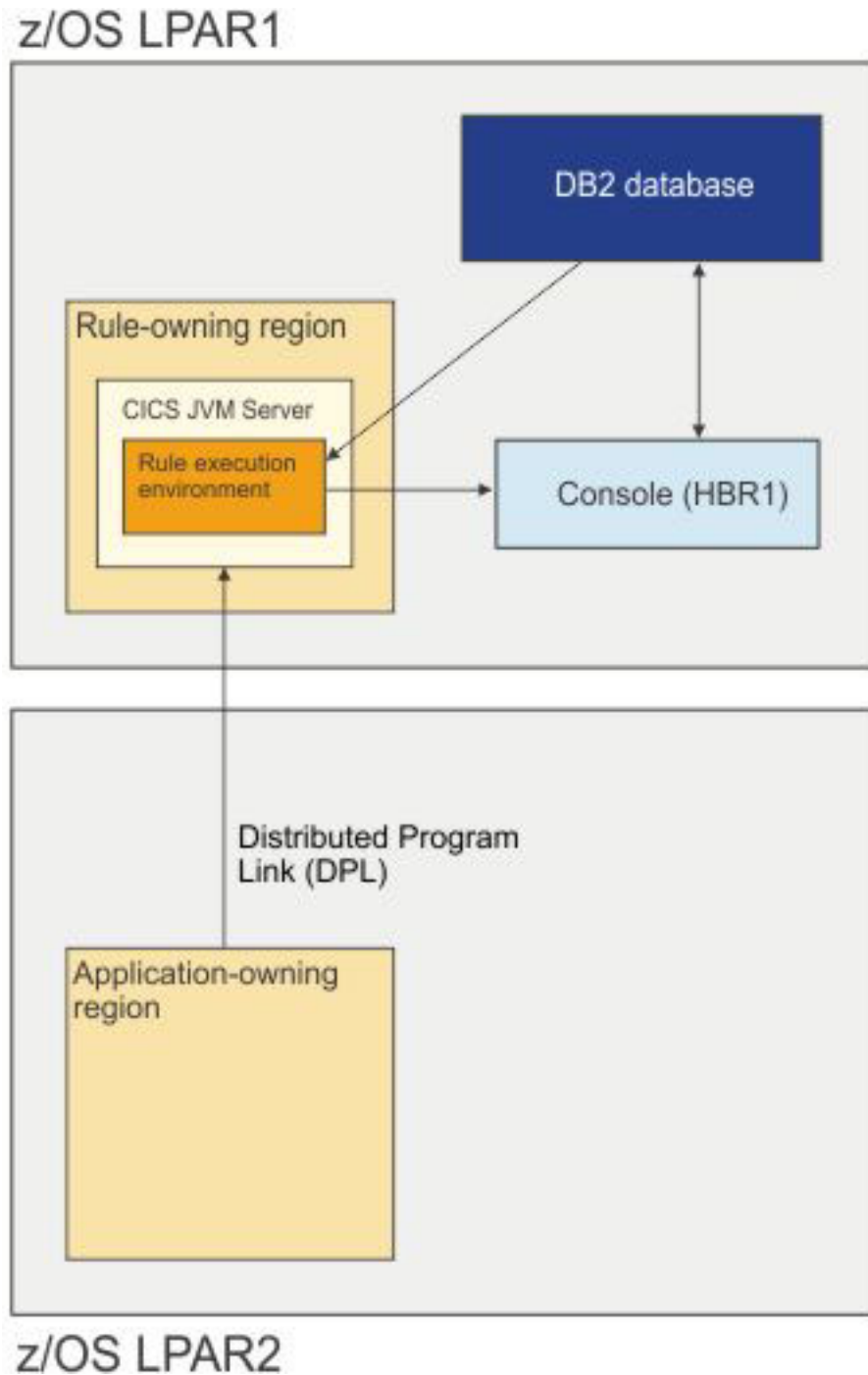
The rule-owning region hosts a zRule Execution Server for z/OS instance that runs locally in a CICS JVM server. The rule-owning region is enabled so that rules applications can start rules in the zRule Execution Server for z/OS instance.

The application-owning region uses a CICS Distributed Program Link (DPL) to run rules in a rule-owning region. The rule-owning region requires a Rule Execution Server console that runs in a separate address space from the regions and has a unique subsystem identifier (HBRSSID). A DB2 database provides the persistence layer. The rule-owning region must be on CICS Transaction Server V4.2 or later. The application-owning region must be on CICS Transaction Server V3.2 or later.

Reasons for choosing this topology

- To run rules in multiple CICS regions if the rules workload does not justify the memory or CPU requirements of running a JVM within every CICS region.
- To run rules in a CICS JVM from a CICS version 3.2 region.
- To connect CICS applications exclusively, and to run rules to the zRule Execution Server for z/OS.
- To ensure the high availability of rules by using either the CICSplex workload manager, or DPL.

Architecture



Topology 4: Operational Decision Manager on WebSphere Application Server for z/OS

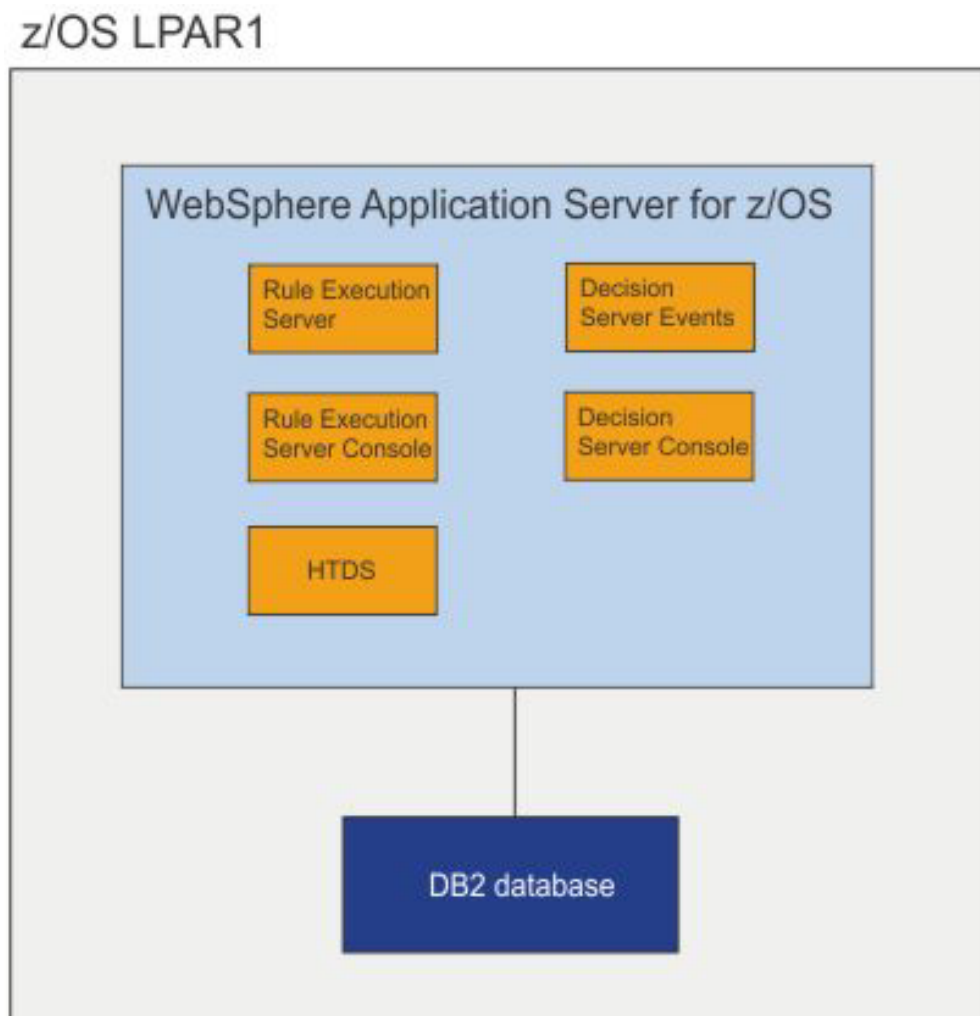
This topology consists of a WebSphere Application Server for z/OS instance that hosts the other components: Rule Execution Server, Rule Execution Server console, Hosted Transparent Decision Services (HTDS), Decision Server Events, and Decision Center console. A DB2 database is used as a persistence layer that stores rule applications and data.

Reasons for choosing this topology

- To call rules from batch, IMS, or CICS applications that are hosted in a WebSphere Application Server for z/OS instance that runs within the same LPAR.
- To use the Decision Server Events component to exploit the event processing capability of Operational Decision Manager.
- To call rules by using a web services interface through the Hosted Transparent Decision Service (HTDS).
- To use the Decision Center consoles for authoring, managing, validating, and deploying business rules and events.

Architecture

The diagram shows a single WebSphere Application Server for z/OS instance with all the available components deployed.



Note: In a WebSphere Application Server for z/OS Network Deployment configuration, the Decision Server Events component cannot be clustered across multiple servers.

Topology 5: batch execution and an embedded zRule Execution Server for z/OS instance

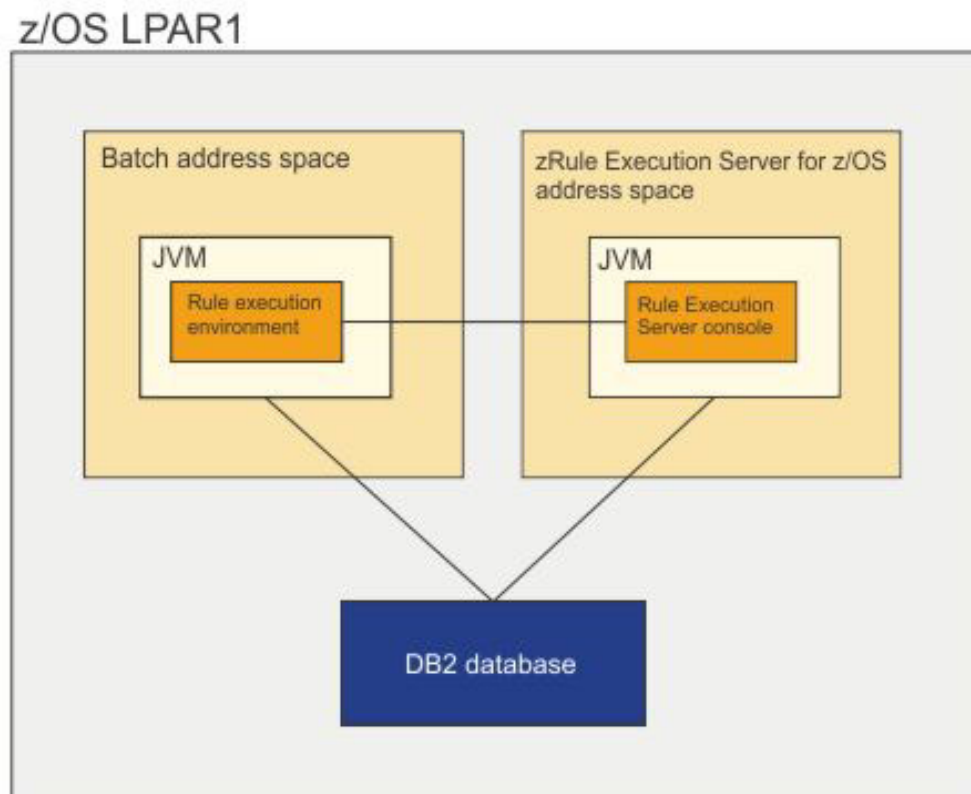
This topology consists of a zRule Execution Server for z/OS instance that is started inside the batch address space. A DB2 database is used as a persistence layer that stores rule applications and data.

Reasons for choosing this topology

To call rules from a Java™ batch application in a zRule Execution Server for z/OS instance that runs in the batch address space.

Architecture

The batch address space starts its own zRule Execution Server for z/OS instance. The Rule Execution Server console runs in a separate zRule Execution Server for z/OS address space.



Java batch executions

The following processes occur when a Java batch application runs rules:

1. The Java batch environment is configured by submitting HBRCJCFG, which takes parameters from ++HBRWORKDS++, SHBRPARAM(*).
2. The HBRMINBJ sample job is used to start a JVM. HBRMINBJ contains all the JVM configuration and adds ++HBRWORKPATH++/config/res to the class path. The generated ra.xml and logging.properties files are found in this directory.
3. The MiniLoanDemo Java sample uses IlrJ2SESessionFactory and standard API calls to invoke rules.

Step 3: Customizing the z/OS configuration and runtime variables

In the SHBRPARM(HBRINST) data set member, customize the z/OS configuration and runtime variables for the topology that you are configuring.

About this task

The z/OS configuration variables are used when generating the working data sets, to initialize the runtime variables that are used by the Operational Decision Management for z/OS JCL. The working data sets contain the customized JCL required to configure the selected topology.

In the SHBRPARM(HBRINST) member, related variables are grouped by execution environment or feature. Customize the variables that are general to all execution environments.

- Customize the CICS variables only if you are using CICS to run your z/OS applications (COBOL or PL/I applications).
- Set the ++DB2HLQ++ variable only if you plan to use this type of persistence.
- If HBRWORKPATH is 35 characters or less in length, the value can be used by all working data sets. However if the value exceeds 35 characters, you must manually update the customized members in SHBRJCL where the value has been inserted into the JCL. If you do not do this, the job might not contain valid JCL when the update job is run to create the working data sets.

Procedure

1. Edit the ++HBRHLLQ++.SHBRPARM(HBRINST) data set member, where ++HBRHLLQ++ represents the high-level qualifier assigned to the data sets during installation.
2. Customize the variables.

Related concepts:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Step 4: Customizing the control statements for your topology

Control statements tell the HBRUPTI job which execution environments to create. When you customize the z/OS configuration and runtime variables, you must customize the control statements for your topology. You specify control statements in the ++HBRHLLQ++.SHBRPARM(HBRCTRL) data set member.

The control statements are written in the following format:

```
<verb> <execution_environment> <ConfigurationVariable1>=<Value1>  
<ConfigurationVariable2>=<Value2>
```

Table 1. Variables used in the control statements

Variable	Description
<verb>	Use CREATE
<execution_environment>	Use ZRES, CICS, WAS, or BATCH

Table 1. Variables used in the control statements (continued)

Variable	Description
<ConfigurationVariable>	<p>Any configuration variable that is specified in HBRINST can be overridden in the control statement.</p> <p>Some execution environments require certain configuration variables to be specified. They are documented for each execution environment.</p>

The control statements can be up to 496 characters in length and can span multiple lines.

Note: The values that are used for control statements must not contain spaces, otherwise the values are truncated.

Related information:

“Step 2: Choosing your topology” on page 2

You can configure Operational Decision Manager for z/OS in different topologies that are based on your production needs and your existing configurations.

“Step 5: Creating the working data sets” on page 14

After you have customized the variables for your site, you must then create the working data sets for your topology.

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

“z/OS runtime variables” on page 126

The runtime variables that you use for setting up the Operational Decision Manager for z/OS runtime are stored in different data set members.

“HBRUPTI job” on page 137

The ++HBRHLQ++,SHBRJCL(HBRUPTI) job creates working data sets based on the control statements that are specified in the ++HBRHLQ++,SHBRPARM(HBRCTRL) data set member.

Customizing topology 1: zRule Execution Server group with a shared console

To configure this topology, control statements are used to create the data sets that are required for each zRule Execution Server for z/OS instance, and the Rule Execution Server console.

Rule execution begins on the first available server in the list. Other servers run rulesets only if rule execution is transferred to them. To route rule execution to a particular server, specify its ID first.

The *HBRA-CONN-SSID* field in the **HBR-CONN-AREA** data area returns the subsystem ID of the server that completed the execution of the ruleset. If a ruleset does not run to completion successfully, this field is set to blanks.

Servers that belong to the same group and are connected to the same Rule Execution Server console share the *HBRWORKPATH* value.

The data sets that are generated enable the zRule Execution Server for z/OS to be called from batch applications and IMS applications.

If you are planning to use this topology to support rule execution from CICS, you must also create a CICS configuration by issuing the following command:

```
CREATE CICS HBRWORKDS=RULES.WORK.REGION1 CICSHLQ=CTS320.CICS650  
CICSCSDDSN=CTS320.REGION2.DFHCSO
```

Variables	Usage notes
<i>HBRMODE</i>	This variable is mandatory. The value is case-sensitive and must be specified in uppercase exactly as shown in the description.
<i>HBRSSID</i>	This variable is mandatory.
<i>HBRWORKDS</i>	This variable is mandatory.
<i>HBRWORKPATH</i>	This variable is optional. Defaults to the value specified in HBRINST.
<i>CICSHLQ</i>	This variable is required for configuring application-owning regions.
<i>CICSCSDDSN</i>	This variable is required for configuring application-owning regions.

The following example creates a single Rule Execution Server console:

```
CREATE ZRES HBRMODE=CONSOLE HBRSSID=HBR1 HBRWORKDS=RULES.WORK.HBR1
```

The following examples create three zRule Execution Server for z/OS instances:

```
CREATE ZRES HBRMODE=RULE HBRSSID=HBR2 HBRWORKDS=RULES.WORK.HBR2
```

```
CREATE ZRES HBRMODE=RULE HBRSSID=HBR3 HBRWORKDS=RULES.WORK.HBR3
```

```
CREATE ZRES HBRMODE=RULE HBRSSID=HBR4 HBRWORKDS=RULES.WORK.HBR4
```

Use these examples and the variables table to create your control statements in `++HBRHLQ++.SHBRPAM(HBRCTRL)`.

The example **CREATE** control statements show that the `++HBRSSIDLIST++` is set to HBR2, HBR3, and HBR4.

Customizing topology 2: one or many CICS rule-owning regions

To configure this topology, control statements are used to generate the data sets for the rule-owning regions and the Rule Execution Server console.

Use the examples and the information in the variables table to create the required control statements in `++HBRHLQ++.SHBRPAM(HBRCTRL)`.

Variable	Usage notes
<i>HBRWORKDS</i>	This variable is mandatory for rule-owning regions, and for the console.
<i>HBRMODE</i>	This variable is mandatory. The value is case-sensitive and must be specified in uppercase exactly as shown.
<i>HBRSSID</i>	This variable is mandatory.
<i>CICSHLQ</i>	This variable is mandatory for rule-owning regions.
<i>CICSCSDDSN</i>	This variable is mandatory for rule-owning regions.
<i>JDBCPLAN</i>	This variable is optional for rule-owning regions.
<i>CICSLIST</i>	This variable is optional for rule-owning regions.

This example creates a rule-owning region:

```
CREATE CICS HBRWORKDS=RULES.WORK.AOR.REGION1 CICSHLQ=CTS420.CICS670
CICSCSDDSN=CTS420.REGION1.DFHCS
```

This example creates a Rule Execution Server console:

```
CREATE ZRES HBRMODE=CONSOLE HBRSSID=HBR1 HBRWORKDS=RULES.WORK.HBR1
```

Customizing topology 3: CICS rule-owning and application-owning regions

To configure this topology, control statements are used to generate the data sets for each rule-owning region, application-owning region and Rule Execution Server console.

Use the examples and the information in the variables table to create the required control statements in `++HBRHLQ++`.SHBRPAM(HBRCTRL).

Variable	Usage notes
<i>HBRMODE</i>	This variable is mandatory. The value is case-sensitive and must be specified in uppercase.
<i>HBRSSID</i>	This variable is mandatory.
<i>HBRWORKDS</i>	This variable is mandatory for application-owning regions, rule-owning regions, and for the Rule Execution Server console.
<i>CICSHLQ</i>	This variable is mandatory for application-owning regions and rule-owning regions.
<i>CICSCSDDSN</i>	This variable is mandatory for application-owning regions and rule-owning regions.
<i>JDBCPLAN</i>	This variable is optional for application-owning regions and rule-owning regions.
<i>CICSLIST</i>	This variable is optional for application-owning regions and rule-owning regions.

The following example creates a rule-owning region:

```
CREATE CICS HBRWORKDS=RULES.WORK.AOR.REGION1 CICSHLQ=CTS420.CICS670
CICSCSDDSN=CTS420.REGION1.DFHCS
```

The following example creates an application-owning region:

```
CREATE CICS HBRWORKDS=RULES.WORK.AOR.REGION2 CICSHLQ=CTS320.CICS650
CICSCSDDSN=CTS320.REGION2.DFHCS
```

The following example creates a Rule Execution Server console:

```
CREATE ZRES HBRMODE=CONSOLE HBRSSID=HBR1 HBRWORKDS=RULES.WORK.HBR1
```

Customizing topology 4: Operational Decision Manager on WebSphere Application Server for z/OS

To configure this topology, control statements are used to create the data sets that are required for the WebSphere Application Server for z/OS instance.

For information on configuring WebSphere Application Server for z/OS in this topology, see the WebSphere Application Server Information Center.

The example control statement creates a working data set by using the variables that are provided in ++HBRHLQ++.SHBRPAM(HBRINST).

Variable	Usage notes
HBRWORKDS	This variable is mandatory.

Use the following example command and the variables table to create your control statements in ++HBRHLQ++.SHBRPAM(HBRCTRL):

```
CREATE WAS HBRWORKDS=RULES.WORK.WAS
```

Customizing topology 5: batch execution and an embedded zRule Execution Server for z/OS instance

To configure this topology, control statements create the data sets that are required for an environment for a batch execution.

Variables	Usage notes
HBRWORKDS	This variable is mandatory.
HBRWORKPATH	This variable is mandatory.

The following example creates a batch execution environment to run the batch job CUSTBILL in Java.

```
CREATE BATCH HBRWORKDS=RULES.WORK.CUSTBILL HBRWORKPATH=/u/rules/workdirs/
custbill
```

It creates the following data sets:

- RULES.WORK.CUSTBILL.SHBRJCL
- RULES.WORK.CUSTBILL.SHBRPAM

You must run `RULES.WORK.CUSTBILL.SHBRJCL(HBRCRTI)` in “Step 6: Creating the working directory” on page 15 to create the work path `/u/rules/workdirs/custbill` for the batch job. `HBRCRTI` is generated by `HBRUUPI`.

Step 5: Creating the working data sets

After you have customized the variables for your site, you must then create the working data sets for your topology.

About this task

In this task, you use the `++HBRHLQ++.SHBRJCL(HBRUUPI)` job to create the working data sets.

Procedure

1. In the `++HBRHLQ++.SHBRJCL(HBRUUPI)` data set member, customize the value of `++HBRHLQ++` which represents the product installation high level qualifier.
2. Ensure that `++HBRHLQ++.SHBRP(HBRINST)` has been configured correctly for your environment and required topologies.
3. Ensure that `++HBRHLQ++.SHBRP(HBRCTRL)` has been configured correctly for your environment and required topologies: For example:

```
CREATE ZRES HBRMODE=CONSOLE HBRSSID=HBR1 HBRWORKDS=RULES.WORK.HBR1
```

creates a single Rule Execution Server console

```
CREATE ZRES HBRMODE=RULE HBRSSID=HBR2 HBRWORKDS=RULES.WORK.HBR2
```

creates a single zRule Execution Server for z/OS instance.
4. Submit `++HBRHLQ++.SHBRJCL(HBRUUPI)`.
5. If an error occurs, check the `HBRUUPI` job output. In particular check the following points:
 - All mandatory variables are set.
 - Variable pairs do not have identical values where this is not permitted.
 - Variables that must be set to integer values are set correctly.
 - Variables that are case-sensitive are set using uppercase, lowercase or mixed case as required.

Results

`HBRUUPI` creates and initializes one or more of the following working data set members depending on your topology choice:

- `SHBRP` members (excluding `HBRINST`), which contain parameters for running zRule Execution Server for z/OS environment as a stand-alone server or as a server running on CICS.
- `SHBRPROC` members, which contain procedures used by started tasks on zRule Execution Server for z/OS
- `SHBRWASC` members, which contain generated properties file for configuring Decision Server Rules , Decision Center consoles and Decision Server Events.
- `SHBRJCL` members, which contain sample JCL for running configuration jobs.

Related concepts:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the `HBRINST` or `HBRCTRL` data set member. To be valid, the values you specify must not contain spaces.

Related information:

“HBRUUPI job” on page 137

The ++HBRHLQ++.SHBRJCL(HBRUUPI) job creates working data sets based on the control statements that are specified in the ++HBRHLQ++.SHBRPARM(HBRCTRL) data set member.

Step 6: Creating the working directory

After you have created the working data sets for your environment, you must create the working directory, which is used to store internal data such as logs. If you are using a file system for your persistence layer, the deployed artifacts are also stored in the working directory.

Before you begin

Ensure that you have specified the location of the working directory using the ++HBRWORKPATH++ JCL variable in the ++HBRHLQ++.SHBRPARM(HBRINST) data set member.

About this task

Create the working directory, which is used to store internal data such as logs. If you are using a file system for your persistence layer, the deployed artifacts are also stored in the working directory.

Note: You need only perform this step once because all servers in the server group share a single working directory.

Procedure

1. Submit the following job:

```
++HBRWORKDS++.SHBRJCL(HBRCRTI)
```

where ++HBRWORKDS++ represents the high-level qualifier assigned to the working data sets that you created in the previous step.
2. Check the output from HBRCRTI to ensure that the job ran successfully.

Step 7: Configuring a DB2 persistence layer

To use a database rather than a file system, configure the persistence layer according to the following steps.

Omit this step if you plan to have the new instance share RuleApps that are stored in the persistence layer of an existing execution environment.

Creating a DB2 database for your rules runtime persistence layer

zRule Execution Server for z/OS and Rule Execution Server can use a DB2 database to store RuleApps and other artifacts.

About this task

The best practice is to use a DB2 database as your persistence type, particularly in a production environment.

Procedure

1. Define a RACF group to control access to the database by completing the following steps:
 - a. Create the `++DB2GROU++` RACF group and add it to an existing superior group by using the following command:
ADDGROUP ++DB2GROU++ SUPGROUP(SYS1) OWNER(SYS1)
 - b. Give the `++DB2GROU++` RACF group access to the DB2 class and the profiles for batch and CICS by using the following commands. The database administrator performs this task.
PERMIT <MY_DB2_SSID>.BATCH CLASS(DSNR) ID(++DB2GROU++) ACCESS(READ)
PERMIT <MY_DB2_SSID>.SASS CLASS(DSNR) ID(++DB2GROU++) ACCESS(READ)

Note: After you enter the PERMIT commands, RACLISTed profiles for DSNR will not reflect the updates until a SETROPTS REFRESH is issued.

2. Create the database tables by running the jobs that are listed in the following table, in the order shown. The data set members are in the `++HBRWORKDS++`.SHBRJCL data set, where `++HBRWORKDS++` represents the high-level qualifier that is assigned to the working data sets.

Note: Verify that the jobs contain the correct settings. The settings were copied from the `++HBRHLQ++`.SHBRPAM(HBRINST) member when you created the working data sets.

Data set member	Description
HBRDSCDB	Creates the table for storing the deployed RuleApps.
HBRDSXOM	Creates the table for storing the deployed Execution Object Models (XOMs).
HBRDSCTR	Creates the trace tables.

3. If you did not run the HBRDSCTR job, edit the `++HBRWORKDS++`.SHBRJCL(HBRDSGRN) job to remove the entries to grant access to those trace tables.
4. Give the `++DB2GROU++` RACF group access to the newly created tables by running the following job:
`++HBRWORKDS++`.SHBRJCL(HBRDSGRN)
5. Give database access to the user ID to be used by the server to connect to the database. Use the following command:
CONNECT ++DB2USER++ GROUP(++DB2GROU++) where `++DB2USER++` represents the user ID used to access the database.

Creating the event runtime database for z/OS

Decision Server Events requires a database for the event runtime. This event runtime is the shared, secured location that contains assets such as business objects, events, and actions.

About this task

The full list of supported database managers is available from Operational Decision Manager Detailed System Requirements.

To install, you must have CREATE and VIEW privileges on the event runtime database.

Procedure

Ask the DBA to use the sample JCL job in `++HBRWORKDS++.SHBRJCL(HBREVCRD)` to create the Events database. The definitions of the database and tables are found in the PDS member `HBREVSQ.L`.

Results

The DB2 event runtime database is now created.

What to do next

Check the results of the job to ensure that the event runtime database has been created.

Creating the Decision Center DB2 database

After you customize the JCL files, you can set up the DB2 database.

Before you begin

Ensure that you have DB2 administrator authority to create a database, and to create and update tables.

About this task

JCL jobs are provided in the `++HBRHLQ++.SHBRJCL` data set to create a DB2 database. You can customize these jobs to meet your requirements. The `HBRUPTI` job creates the instance of the server and configures the jobs with the values defined in the `HBRINST` member. For more information, see *Customizing the z/OS configuration and runtime variables*.

Note:

You must customize the variables, as indicated in the comments within each PDSE member, before you run the `HBRUPTI` job on the z/OS system. Customize the JCL either by using a manual search and replace, or by using an ISPF macro. Your customized variables are used to create your new data sets, so it is important to review your variable values before you run any job.

After you customize the JCL variables in your initial configuration, you can submit the jobs to create a database for Decision Center.

Procedure

1. If you are using RACF, configure RACF security for the DB2 database by adding the `++DB2GROUP++` to an existing superior group by using the following command:

```
ADDGROUP ++DB2GROUP++ SUPGROUP(SYS1) OWNER(SYS1)
```

Where `++DB2GROUP++` represents the name of the RACF security group that is provided to you by your security administrator.

2. Run the `HBRDCCDB` JCL job to create the database artifacts for Decision Center. The database administrator typically performs this task.

The job is in the `++HBRWORKDS++.SHBRJCL` data set. `++HBRWORKDS++` represents the high-level qualifier that is assigned to the working data sets.

Note:

Verify that the JCL jobs contain the correct settings for your site. The settings were copied from the `++HBRHLQ++.SHBRP` member when you created the working data sets.

3. Give the RACF `++DB2GROU` access to the created tables by running `++HBRWORKDS++.SHBRJCL`.
4. Add the user ID to the `++DB2GROU` to connect to the database by submitting the following command:

```
CONNECT ++DB2USER++ GROUP(++DB2GROU++)
```

Where `++DB2USER++` is the user ID to access DB2.

Step 8: Configuring your topology

When you have decided on the topology that meets your requirements, configure it by following the relevant instructions.

Configuring topology 1: zRule Execution Server group with a shared console

This topology consists of a server group that contains multiple instances. The topology also contains a shared Rule Execution Server console. Each component runs in a separate address space. A DB2® database is the persistence layer that stores rule applications and data.

Configuring a zRule Execution Server for z/OS server group

Configure a group of zRule Execution Server for z/OS instances to execute rules for z/OS applications (COBOL or PL/I applications) running as batch jobs.

Before you begin

Before you begin, you must complete the initial configuration tasks that are described in “Configuring,” on page 1 (authorize the load library, customize the variables, decide on a topology, create the working data sets and directory, configure DB2, and configure security).

About this task

Complete the following steps to configure a group of 1 to 32 zRule Execution Server for z/OS instances with a shared console.

For information about topology configuration options, see “Step 2: Choosing your topology” on page 2.

In the following procedure, the server group is defined by the `++HBRSSIDLIST++` JCL variable.

Procedure

1. Define a new subsystem for each server instance to run in. The system programmer performs this task by using the following command:

```
SETSSI ADD, SUBNAME=<HBRSSID_NAME>
```

If the command completes successfully, the following message is displayed:
IEFJ022I SETSSI ADD COMMAND FOR SUBSYSTEM <HBRSSID_NAME> COMPLETED SUCCESSFULLY.

Consider adding a definition to the system parameter library (parmlib) member IEFSSNxx to make this permanent. For example:

```
SUBSYS SUBNAME (<HBRSSID_NAME>)
```

2. Enable each server instance to run as a started task. For each server instance, copy the following data sets to the SYS1.PROCLIB data set or to a suitable equivalent. The system programmer performs this task.

- a. Copy the ++HBRWORKDS+.SHBRPROC(HBRXMSTR) data set member of each server instance to SYS1.PROCLIB(<HBRSSID_NAME>MSTR).

For example, to enable as started tasks server instances HBR1 and HBR2, copy HBR.WORKDS.HBR1.SHBRPROC(HBRXMSTR) to SYS1.PROCLIB(HBR1MSTR) and HBR.WORKDS.HBR2.SHBRPROC(HBRXMSTR) to SYS1.PROCLIB(HBR2MSTR).

- b. Authorize the MSTR member for each server. Use the following commands:

```
RDEFINE STARTED <HBRSSID_NAME>MSTR.*  
STDATA(USER(<HBRSSID_USER>)GROUP(<HBRSSID_GROUP>))  
SETR REF RACLIST(STARTED)
```

where <HBRSSID_USER> represents the server user ID and <HBRSSID_GROUP> represents the name of the RACF security group that is provided to you by your security administrator.

Configuring CICS to execute rules on zRule Execution Server for z/OS

After you configure a zRule Execution Server for z/OS server group, you can optionally configure CICS to execute rules on a server instance.

Before you begin

Verify that you meet the following prerequisites:

- You have configured a zRule Execution Server for z/OS server group as described in “Configuring topology 1: zRule Execution Server group with a shared console” on page 18.
- In “Step 3: Customizing the z/OS configuration and runtime variables” on page 9, you have customized the CICS-related JCL variables listed in Configuration variables for configuring CICS to execute rules on a zRule Execution Server for z/OS instance.

See “Step 2: Choosing your topology” on page 2 for information about topology configuration options.

About this task

Use the following procedure to configure CICS to execute rules on zRule Execution Server for z/OS.

You can optionally configure CICS to execute the **HBRC** transaction at startup to connect to the server instance automatically, if it is running. With this option, CICS is immediately ready to execute rules. If the server is not running, CICS starts up normally but you must execute the HBRC transaction manually before rule execution can begin.

Note: This configuration is supported for CICS version 3.2 or later. If you are using CICS 4.2 or later, consider running zRule Execution Server for z/OS inside the CICS JVM server. For more information, see “Configuring a CICS rule-owning region to execute rules in a CICS JVM server” on page 29.

In the following procedure, the server group is defined by the ++HBRSSIDLIST++ JCL variable.

Procedure

1. Submit the ++HBRWORKDS++.SHBRJCL(HBRCSO) job to define the resources required by CICS.
2. In the CICS system initialization table, add the list name represented by ++CICSLIST++ to the list of resource definition groups specified by the **GRPLIST** parameter.
3. Edit the CICS JCL to include the zRule Execution Server for z/OS data sets:
 - a. Ensure that the ++HBRHLQ++.SHBRICSO data set is added to the **DFHRPL** concatenation.
 - b. Pass the necessary runtime variables to the server by adding the SHBRPAM(HBRICSO) and SHBRPAM(HBRMMN) data set members to the **HBRENVPR** DD statement as shown in the following DD statements:

```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPAM(HBRICSO)
//          DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPAM(HBRMMN)
```
 - c. If you plan to run the Miniloan sample application to verify your configuration, add the SHBRPAM(HBRSCEN) data set member to the SCENARIO DD statement. The HBRSCEN member contains input values to the Miniloan application.

```
//SCENARIO DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPAM(HBRSCEN)
```
4. Start the zRule Execution Server for z/OS and the Rule Execution Server console.
5. Start CICS and install the **HBRGROUP** resources using the following command:
CEDA INSTALL GROUP(HBRGROUP)
6. Test the configuration by running the **HBR** transaction to connect CICS to zRule Execution Server for z/OS. If CICS succeeds in connecting to the server, message GBRZC9001I is issued. If CICS fails to connect to the server, message GBRZC9000E is issued with a return code containing diagnostic information.
7. Optional: To configure CICS to connect automatically at startup to the running server instance, take either of the following actions:
 - If you do not have a program list table defined, add the **PLTPI=HB** parameter to the CICS system initialization table.
 - If you already have a program list table defined and specified in your CICS system initialization table, add the **HBRCCON** program to the list by including the following statement: DFHPLT TYPE=ENTRY, PROGRAM=HBRCCON.
8. Optional: You can verify your configuration by running the Miniloan sample application as a CICS application. For more information see Sample: running the Miniloan application as a CICS application.

Configuring IMS to execute rules on zRule Execution Server for z/OS

After you configure a zRule Execution Server for z/OS server group, you can optionally configure IMS to execute rules on a server instance.

Before you begin

- Configure a zRule Execution Server for z/OS server group as described in “Configuring topology 1: zRule Execution Server group with a shared console” on page 18.

- In “Step 3: Customizing the z/OS configuration and runtime variables” on page 9, customize the JCL variables that are listed in Configuration variables for configuring IMS to run rules on a zRule Execution Server for z/OS instance.
- zRule Execution Server for z/OS supports IMS versions 11, 12, and 13.
- IMS applications that uses zRule Execution Server for z/OS for rules checking can run in Message Processing Regions (MPRs), Batch Message Processing (BMP) regions, or DLIBATCH regions.
- IMS and Decision Server for z/OS must run in the same logical partition (LPAR).

See “Step 2: Choosing your topology” on page 2 for information about topology configuration options.

About this task

After you configure a zRule Execution Server for z/OS server group, you can configure IMS to execute rules on a server instance.

Procedure

1. Make the following changes to the execution JCL:
 - a. Add an HBRENVPR DD statement that specifies the zRule Execution Server for z/OS instance to be called by the IMS application. For example:


```
//HBRENVPR DD DISP=SHR,
// DSN=++HBRWORKDS++.SHBRPAM(HBRBATCH)
```
 - b. In the STEPLIB of the execution JCL, you must concatenate the SHBRLOAD library. For example:


```
// DD DSN=++HBRHLQ++.SHBRLOAD,DISP=SHR
```

Note: For optimum performance, consider putting the load library at the top of the STEPLIB or in the pageable link pack area (PLPA). For more information, see the Choosing IMS options for performance section in the IMS information center.

2. If you are using IMS message processing regions (MPRs), complete the following step:
 - Edit the DFSINTxx member of IMS.PROCLIB for your MPR region. Add the preinitialization routine **HBRIPREI**. The HBRIPREI preinitialization routine establishes the connection between the MPR and the zRule Execution Server for z/OS.
 - Code the **PREINIT** parameter of DFSMPR as PREINIT=xx, where xx is the two-character suffix of the DFSINTxx PROCLIB member.

If you are using MPR, you must still include HBRCON and HBRDISC calls in your Message Processing Program (MPP); they do not affect the MPR connection to zRule Execution Server for z/OS but can be used for setting up other services if required.

3. Restart IMS to register the changes. zRule Execution Server for z/OS must be started before IMS when it uses IMS Message Processing Regions. If IMS is started before zRule Execution Server for z/OS, then restart the IMS Message Processing Region so that the HBRIPREI pre-initialization routine can initiate a connection with zRule Execution Server for z/OS.
4. Aside from the execution JCL changes (mentioned above), the use of the zRule Execution Server for z/OS from an IMS application program does not require any other setup in IMS. There are no special definitions in the IMS System definition nor in the IMS startup parameters. Everything else in IMS is business

as usual. For example, if the COBOL program that uses zRule Execution Server for z/OS is a BMP, then the definition in the IMS system definition for the BMP does not change. In addition, there are no changes in the PSB that is associated with the program.

Securing zRule Execution Server for z/OS resources

You can optionally control access to zRule Execution Server for z/OS resources, files, and functions by using RACF.

Security options:

You can secure all zRule Execution Server for z/OS resources or only specific ones. You also have the option of leaving resources unsecured. The appropriate security level depends on how you use the server.

If you plan to run your zRule Execution Server for z/OS instance in a production environment, you might want to secure some or all of your zRule Execution Server for z/OS resources. On the other hand, if you plan to run your server instance in a development or test environment, you might decide to leave resources unsecured.

Note: If you use a security product other than RACF, bypass the following instructions and issue the equivalent security commands for your product.

Within the file system, you can secure the following resources:

- The working directory so that only authorized user IDs can access internal data.
- The installation directory so that only authorized user IDs can access the files needed to run the server.

Using RACF, you can secure the following resources:

- The server resources you use to perform the following tasks:
 - Issue zRule Execution Server for z/OS commands from the z/OS console (or equivalent).
 - Sign into the Rule Execution Server console.
 - Connect to zRule Execution Server for z/OS to execute rulesets.
- A subset of server resources. For example, you can secure access to the Rule Execution Server console only.

You also have the option of disabling security or running the server without security. The topics that follow explain how to configure support for the different options.

Securing access to the working directory and the installation directory:

You can secure access to the working directory and the installation directory for the server instance so that only authorized user IDs can access them.

The working directory, represented by the `++HBRWORKPATH++` JCL variable, holds data such as installation logs, component details, and trace output. The installation directory, represented by the `++HBRINSTPATH++` variable, contains files needed to run the server instance.

Give the server user ID the following permissions:

- Read, write and execute permissions to the `++HBRWORKPATH++` directory.
- Read and execute permissions to the `++HBRINSTPATH++` directory.

Related information:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Managing server security:

You can enable or disable security for zRule Execution Server for z/OS instances or specific server resources.

Creating the RACF classes for securing server resources:

Create and activate the RACF classes that you need to secure the server resources for your zRule Execution Server for z/OS server.

About this task

You manage security for zRule Execution Server for z/OS using RACF classes. When you create and activate the HBRADMIN class, server security is enabled for all zRule Execution Server for z/OS instances that are running on the z/OS system. Users can then be authorized for certain tasks by using the different classes.

Procedure

1. Run the ++HBRWORKDS++.SHBRJCL(HBRCRECL) job for every server in the server group. The table shows the RACF classes that are created by the JCL.

Table 2. RACF classes for server security

Class	Description
HBRADMIN	Controls whether server security and security for specific server resources are enabled or disabled.
HBRCONN	Specifies the user IDs that are authorized to connect to the zRule Execution Server for z/OS and execute rule sets. This class is ignored if server security is disabled.
HBRCMD	Specifies the user IDs that are authorized to issue zRule Execution Server for z/OS commands such as START, STOP, PAUSE, or RESUME from the z/OS console (or equivalent). This class is ignored if server security is disabled.

Note: The supplied JCL gives each class a POSIT value of 128. You must customize this POSIT value for your own local security setup. More information on POSIT values can be found in *Security Server RACF Security Administrator's Guide*.

2. If the BPX.SERVER FACILITY class profile is defined, give the server user ID read access to the class profile by using the following commands:

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(<HBRSSID_USER>) ACCESS(READ)  
SETROPTS RACLIST(FACILITY) REFRESH
```

where *HBRSSID_USER* represents the server ID under which the server runs.

Disabling types of security:

You can disable all security for a zRule Execution Server for z/OS server or only specific types of security.

About this task

When you create and activate the HBRADMIN RACF class, server security is enabled for all zRule Execution Server for z/OS servers running on the z/OS system.

In some cases, you might want to disable server security for one or more servers. For example, you might not want to secure servers in a development or test environment. Alternatively, you might want to disable certain types of security for a server while maintaining other types of security. For example, you might want to control access to the Rule Execution Server console but allow any application to connect to the server.

To disable a type of security for a server, add the corresponding resource profile to the HBRADMIN class. The following table lists the resource profiles and the types of security they disable:

Resource profile	Description
<HBRSSID_NAME>.NO.SUBSYS.SECURITY	Disables all security for a particular server. If server security is disabled, HBRCONN and HBRCMD classes are not used.
<HBRSSID_NAME>.NO.CONNECT.SECURITY	Disables connection security for a particular server, but maintains other types of security.
<HBRSSID_NAME>.NO.COMMAND.SECURITY	Disables command security for a particular server, but maintains other types of security. Note: If you disable command security, any user can issue zRule Execution Server for z/OS commands from the z/OS console.

To disable security for a single server, set <HBRSSID_NAME> to a subsystem ID in the ++HBRSSIDLIST++ JCL variable. To manage server security for multiple servers or for the entire server group, specify a wildcard as the subsystem ID.

If you are managing multiple servers, you can use a wildcard as described previously to specify the subsystem ID, *HBRSSID_NAME*, in the resource profile referred to in the following procedure.

Procedure

1. Add the resource profile to the HBRADMIN class by using the following command:

```
RDEFINE HBRADMIN <RESOURCE_PROFILE> UACC(NONE)
```

where <RESOURCE_PROFILE> represents one of the resource profiles that are listed in the previous table.

2. Refresh the HBRADMIN class by using the following command:

```
SETROPTS RACLIST(HBRADMIN)REFRESH
```

Related information:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Managing connection security:

Use connection security to ensure that only authorized user IDs are allowed to start a zRule Execution Server for z/OS or connect to execute rulesets.

About this task

Connection security uses the HBRCONN RACF class to authorize user IDs to start or connect to the server. However, if the HBRADMIN class includes either of the following resource profiles, connection security for the server is disabled and the HBRCONN class is ignored.

- `<HBRSSID_NAME>.NO.SUBSYS.SECURITY`
- `<HBRSSID_NAME>.NO.CONNECT.SECURITY`

To manage connection security for a single server, set *HBRSSID_NAME* to a subsystem ID in the `++HBRSSIDLIST++` JCL variable. To manage connection security for multiple servers or for the entire server group, specify a wildcard as the subsystem ID.

Note: In some cases, you might want to disable connection security but maintain the other types of security. For more information, see “Disabling types of security” on page 24.

To implement connection security, authorize the user ID under which the server runs and the user IDs of any applications that execute rulesets.

If you are managing multiple servers, you can use a wildcard as described previously to specify the subsystem ID, *HBRSSID_NAME* in the following procedure.

Procedure

1. Authorize the server user ID:
 - a. Create the resource profile, *HBRSSID_NAME*, in the HBRCONN RACF class.
Use the following command:
RDEFINE HBRCONN <HBRSSID_NAME> UACC(NONE)
 - b. Give the server user ID UPDATE access to the *HBRSSID_NAME* resource profile.
Use the following command:
**PERMIT <HBRSSID_NAME> CLASS(HBRCONN) ID(<HBRSSID_USER>)
ACCESS(UPDATE)**
where *<HBRSSID_USER>* represents the server user ID.

Note: zRule Execution Server for z/OS fails to initialize if *<HBRSSID_USER>* lacks UPDATE access to this profile.

- c. Refresh the *HBRSSID_NAME* resource profile by using the following command:
SETROPTS RACLIST(HBRCONN) REFRESH

2. Authorize the applications:

- a. Give READ access to the *HBRSSID* resource profile to each user ID you want to authorize.

Use the following command:

```
PERMIT <HBRSSID_NAME> CLASS(HBRCONN) ID(<USER_ID>) ACCESS(READ)
```

Note: For batch jobs, *<USER_ID>* is the RACF user ID used by the batch job. For CICS transactions, *<USER_ID>* is the user ID assigned to the CICS address space.

- b. Refresh the *HBRSSID_NAME* resource profile by using the following command:

```
SETROPTS RACLIST(HBRCONN) REFRESH
```

Related information:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Managing console security:

Use console security to ensure that only authorized user IDs can access the Rule Execution Server console.

About this task

Console security controls the ability to sign into the Rule Execution Server console. If console security is enabled, users must enter a user ID and password to sign in. If the HBRADMIN class includes either of the following resource profiles, console security for the server instance is disabled and any other console-related resource profiles in the HBRADMIN class are ignored:

- *<HBRSSID_NAME>.NO.SUBSYS.SECURITY*

To manage console security for a single server, set *<HBRSSID_NAME>* to a subsystem ID in the server group that is specified by the *++HBRSSIDLIST++* JCL variable. To manage console security for multiple servers or for the entire server group, specify a wildcard as the subsystem ID.

Note: In some cases, you might want to disable console security but maintain the other types of security. For more information, see “Disabling types of security” on page 24.

The following table lists the profiles and the roles they represent. Roles are listed in order of increasing authority.

Resource profile	Role description
<i><HBRSSID_NAME>.ROLE.RESMON</i>	Users with monitoring rights are only allowed to view and explore RuleApps, rulesets, decision services, execution units, and statistics. They are not allowed to modify them. They can also select a trace configuration and view and filter trace information in Decision Warehouse (applies only to Rule Execution Server on WebSphere Application Server for z/OS).

Resource profile	Role description
<HBRSSID_NAME>.ROLE.RESDEP	In addition to monitoring rights, users with deploying rights are allowed to deploy RuleApp archives and to edit and remove entities (RuleApps, rulesets, decision services, Java Execution Object Model (XOM) resources and libraries), and run diagnostics.
<HBRSSID_NAME>.ROLE.RESADMIN	Users with administrator rights have full control over the deployed resources and access to information on the server. They can carry out the following actions: <ul style="list-style-type: none"> • Deploy, browse, and modify RuleApps, Java XOM resources, and libraries. • Monitor the decision history, purge, and back up the history. • Select a trace configuration, view and filter trace information, and clear trace information in Decision Warehouse. • Run diagnostics and view server information.

Procedure

1. Define each resource profile that is shown in the previous table to the HBRADMIN class by using the following command:
RDEFINE HBRADMIN <RESOURCE_PROFILE> UACC(NONE)
2. Assign each user ID to one of the resource profiles by using the following command:
PERMIT <RESOURCE_PROFILE> CLASS(HBRADMIN) ID(<USER_ID>) ACCESS(UPDATE)
3. Refresh the HBRADMIN class by using the following command:
SETROPTS RACLIST(HBRADMIN) REFRESH

Related information:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Managing command security:

Use command security to ensure that only authorized users can issue zRule Execution Server for z/OS commands from the z/OS console.

About this task

Command security uses the HBRCMD RACF class to authorize user IDs to issue zRule Execution Server for z/OS commands. However, if the HBRADMIN class includes either of the following resource profiles, command security for the server instance is disabled and the HBRCMD class is ignored:

- <HBRSSID_NAME>.NO.SUBSYS.SECURITY
- <HBRSSID_NAME>.NO.COMMAND.SECURITY

To manage command security for a single server, set `<HBRSSID_NAME>` to a subsystem ID in the server group that is specified by the `++HBRSSIDLIST++` JCL variable. To manage command security for multiple servers or for the entire server group, specify a wildcard as the subsystem ID.

Note: In some cases, you might want to disable command security but maintain the other types of security. For more information, see “Disabling types of security” on page 24.

To enable zRule Execution Server for z/OS command security, define a resource profile to the HBRCMD class for each command you want to secure. Then, authorize each user ID individually to issue each command. For example, you must separately authorize a user ID to issue the **PAUSE** command and the **RESUME** command.

The following table lists the resource profiles and the commands that they control.

Resource profile	Command
<code><HBRSSID_NAME>.SET.TRACE</code>	SET TRACE
<code><HBRSSID_NAME>.PAUSE</code>	PAUSE
<code><HBRSSID_NAME>.RESUME</code>	RESUME

If you are managing multiple servers, you can use a wildcard as described previously to specify the subsystem ID, `HBRSSID_NAME`, in the resource profile referred to in the following procedure.

Procedure

1. Define to the HBRCMD class each resource profile that is shown in the previous table.

Use the following command:

```
RDEFINE HBRCMD <RESOURCE_PROFILE> UACC(NONE)
```

2. Authorize each user ID to issue each command.

Use the following RACF command:

```
PERMIT <RESOURCE_PROFILE> CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
```

where `<RESOURCE_PROFILE>` is a resource profile that is listed in the previous table.

The following example shows how to authorize user ID, `SYSPROG1`, to issue the **SET TRACE** command:

```
PERMIT <HBRSSID_NAME>.SET.TRACE CLASS(HBRCMD) ID(SYSPROG1)  
ACCESS(UPDATE)
```

3. Refresh the HBRCMD class.

Use the following command:

```
SETROPTS RACLIST(HBRCMD) REFRESH
```

Related information:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Commands

Starting a new server instance

Start one or more newly-created zRule Execution Server for z/OS instances.

About this task

Start a zRule Execution Server for z/OS instance and open the Rule Execution Server console.

Note: To start a server instance running on CICS JVM server, start your CICS system instead.

Procedure

1. At the z/OS console (or equivalent) command line, issue the command, `START <HBRSSID_NAME>MSTR`, where `HBRSSID_NAME` represents a subsystem ID in the server group defined by the `++HBRSSIDLIST++` JCL variable. Repeat the command for each additional server in `++HBRSSIDLIST++`.
2. Ensure that one of the servers in the `HBRSSIDLIST` has the `HBRMODE` set to `CONSOLE`, then display the Rule Execution Server console by opening a browser and navigating to `http://<hostname>:++HBRCONSOLEPORT++/res`, where `hostname` is the TCP/IP name of the z/OS machine where you are running zRule Execution Server for z/OS and `++HBRCONSOLEPORT++` is the HTTP port you specify to access the console from your web browser.
3. Sign in to the console using your zRule Execution Server for z/OS user ID and password.

If you are unable to sign in, see [Troubleshooting zRule Execution Server for z/OS](#).

Results

You have successfully configured zRule Execution Server for z/OS and started a server instance. For more information on administering the server, see [Administering zRule Execution Server for z/OS](#).

What to do next

You can now develop z/OS applications (COBOL or PL/I applications) to execute rules on zRule Execution Server for z/OS. For more information, see [Developing applications](#).

Configuring topologies 2 and 3: CICS rule and application-owning regions

Topology 2 consists of a zRule Execution Server for z/OS console running in a separate address space, and the execution units (XU) deployed into the JVM server (or JVM servers) in a single CICS region. Topology 3 exploits two types of CICS region (the rule-owning region and the application-owning region) that can execute rules.

Configuring a CICS rule-owning region to execute rules in a CICS JVM server

After you configure a zRule Execution Server for z/OS server group, you can optionally configure the server group to run on a CICS JVM server.

Before you begin

Verify that you meet the following prerequisites:

- You have configured a zRule Execution Server for z/OS server group as described in “Configuring topology 1: zRule Execution Server group with a shared console” on page 18.
- In “Step 3: Customizing the z/OS configuration and runtime variables” on page 9, you have customized the CICS-related JCL variables in Configuration variables for configuring zRule Execution Server for z/OS running on a CICS JVM server.

`++HBRWORKDS++` represents the high-level qualifier assigned to the working data sets and `++HBRHLQ++` represents the high-level qualifier assigned to the data sets during installation.

About this task

Use the following procedure to configure the zRule Execution Server for z/OS to run on a CICS JVM server with CICS version 4.2 or later.

For information about topology options, see “Step 2: Choosing your topology” on page 2.

You can configure CICS JVM Server as a highly available, scalable environment using standard CICSplex System Management (CPSM) capabilities. In this configuration, the Task Related User Exit (TRUE) calls the HBRCJVMS program using the **EXEC CICS LINK** command, which is routed to the specified CICS region. To create this configuration, you must perform the following procedure on both the calling CICS JVM server and the eventual rule execution CICS regions. For more information about CPSM, see CICS Transaction Server for z/OS information center.

In the following steps, the server group is defined by the `++HBRSSIDLIST++` JCL variable.

Procedure

1. Submit the `++HBRWORKDS++.SHBRJCL(HBRCRTI)` job to create a working directory for the server on CICS. The `++HBRWORKPATH++` JCL variable specifies the location of the working directory.
2. Run the following job to create the default CICS JVM profile in the working directory specified by the `++HBRWORKPATH++` JCL variable:
 - Submit the `++HBRWORKDS++.SHBRJCL(HBRCJVMP)` job to create the profile, HBRJVM.
3. In the CICS system initialization table, look for the variable, **JVMPROFILEDIR**, which points to the JVM profiles directory as described in the CICS V4.2 information center at CICS Transaction Server for z/OS, Version 4 Release 2. If this parameter is not listed, define a new **JVMPROFILEDIR** to store your JVM profile.
4. Copy the profile HBRJVM from the location specified by the `++HBRWORKPATH++` variable to the **JVMPROFILEDIR** directory for your CICS region. Ensure that the profile has appropriate read permissions set so that CICS can open it. If you are using the JVM server for another purpose you must merge the generated profile with the preexisting one. Otherwise, you can replace any existing profile file with the generated one.
5. Define the CICS resources required by the server:

- a. Submit the `++HBRWORKDS++.SHBRJCL(HBRCSD)` job to define resources required by all CICS versions.
 - b. Review the `++HBRWORKDS++.SHBRJCL(HBRCSDJ)` job and make any required changes to the DB2 configuration information.
 - c. Submit the job.
6. In the CICS system initialization table, add `++CICSLIST++` to the list of resource definition groups specified by the **GRPLIST** parameter.
 7. Edit the CICS region startup JCL to include the zRule Execution Server for z/OS data sets:
 - a. If you are using a DB2 database, ensure that the STEPLIB for your CICS started task includes the DB2 load data sets, SDSNLOAD2 and SDSNLOAD.
 - b. Ensure that the `++HBRHLQ++.SHBRICIS` data set is added to the DFHRPL concatenation.
 - c. Pass runtime variables to the server by adding the following SHBRPARM data set members to the HBRENVPR DD statement. The HBRCICSD data set member includes necessary DB2 runtime variables.
The following JCL shows the member names:


```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCISJ)
//          DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCMMN)
//          DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCICSD)
```
 - d. If you plan to run the Miniloan sample application to verify your configuration, add the following DD statement. The HBRSCEN data set member contains input values to the Miniloan application.


```
//SCENARIO DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRSCEN)
```
 8. Start the zRule Execution Server for z/OS and the Rule Execution Server console.
 9. Start CICS and install the HBRGROUP resources by using the following command:


```
CEDA INSTALL GROUP(HBRGROUP)
```
 10. If you are using a database, connect the DB2CONN resource to the database by completing the following steps:
 - a. Issue the following CICS command to display the active definition of the DB2CONN resource:


```
CEMT INQUIRE DB2CONN
```
 - b. Change the CONNECTST property from Notconnected to Connected.
 - c. If you want CICS to connect to the database automatically at start up, set the system initialization table (SIT) parameter, **DB2CONN=YES**.
 11. Test the configuration by running the **HBRC** transaction to connect CICS to zRule Execution Server for z/OS.
If CICS succeeds in connecting to the server, message GBRZC9001I is issued. If CICS fails to connect to the server, message GBRZC9000E is issued with a return code containing diagnostic information.
To disconnect from the server, run the **HBRD** transaction.
See messages for details.
 12. Use the console to view the CICS JVM server instance where the zRule Execution Server for z/OS is running.
 13. Optional: You can verify your configuration by running the Miniloan sample application as a CICS application. For more information see Sample: running the Miniloan application as a CICS application.

Configuring a CICS application-owning region to execute rules in a CICS rule-owning region

You can configure an application-owning region to execute rules in a remote rule-owning region. The connection uses a distributed programming link (DPL). The Rule Execution Server runs in the rule-owning region on a CICS JVM.

Before you begin

Ensure that the rule-owning region is configured. For more information, see “Configuring a CICS rule-owning region to execute rules in a CICS JVM server” on page 29. Also define the required CICS resources (CONNECTION, IPCONN, SESSION) for the DPL, and enable interregion communication (IRC) on both the application-owning region and on the rule-owning region.

For information on defining CICS resources and enabling IRC, see How to define connections to remote systems in the CICS Transaction Server information center.

If you are configuring the application-owning region on another MVS™ image or sysplex you must copy the client libraries to the MVS image.

The application-owning region must be on CICS Transaction Server V3.2, or later, and must be connecting to a rule-owning region on CICS Transaction Server V4.1, or later.

About this task

To complete this task, you copy the client libraries to the application-owning region, edit the required variables and submit the JCL jobs. You can then start CICS with a **CEDA** command, install the required CICS resources, and test the connection between the application-owning region and the rule-owning region by running the HBRC transaction. You must run the HBRC transaction each time the CICS region is cycled to initialize the connection to the rule-owning region.

Procedure

1. Complete this step if you are configuring the application-owning region on another MVS image or sysplex. Copy the client libraries to the MVS image of the application-owning region. Firstly copy the ++HBRHLQ++.SHBRICIS dataset, then copy the following members:
 - ++HBRWORKDS++.SHBRJCL(HBRCSO)
 - ++HBRWORKDS++.SHBRPAM(HBRICSO)
 - ++HBRWORKDS++.SHBRPAM(HBRMMN)
 - ++HBRWORKDS++.SHBRPAM(HBRSCN)
2. Edit the ++HBRWORKDS++.SHBRJCL(HBRCSO) dataset so that the HBRCJVM program is defined as a remote server program. You can uncomment and configure the sample remote program and delete or comment out the original definition. For information on how to do this, see Defining remote server programs in the CICS Transaction Server information center.
3. Submit the ++HBRWORKDS++.SHBRJCL(HBRCSO) job so that the resources required by CICS are defined.
4. In the CICS system initialization table, add the list name that is represented by ++CICSLIST++ to the list of resource definition groups that is specified by the **GRPLIST** parameter.
5. Edit the SHBRPAM(HBRICSO) parm member. Change the *HBRTARGETRES* variable to RCICSJVM.

6. Edit the CICS JCL to include the zRule Execution Server for z/OS data sets:
 - a. Ensure that the ++HBRHLQ++,SHBRICS data set is added to the DFHRPL concatenation.
 - b. Pass the necessary runtime variables to the server by adding the SHBRPDM(HBRICSZ) and SHBRPDM(HBRMNM) data set members to the HBRENVPR DD statement as shown in the following DD statements:


```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPDM(HBRICSZ)
//          DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPDM(HBRMNM)
```
 - c. If you plan to run the Miniloan sample application to verify your configuration, add the SHBRPDM(HBRSCEN) data set member to the SCENARIO DD statement. The HBRSCEN member contains input values for the Miniloan application.


```
//SCENARIO DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPDM(HBRSCEN)
```
7. Start CICS.
8. Install the HBRGROUP resources by using the following command:


```
CEDA INSTALL GROUP(HBRGROUP)
```
9. Test the configuration by running the HBRC transaction to initialize the application-owning region. If CICS succeeds in connecting to the server, message GBRZC9001I is issued. If CICS fails to connect to the server, message GBRZC9000E is issued with a return code containing diagnostic information. You must run the HBRC transaction each time the CICS region is cycled to initialize the connection to the rule-owning region.
10. Optional: Configure CICS to connect to the server instance when CICS starts by doing one of the following:
 - If you do not have a program list table defined, add the **PLTPI=HB** parameter to the CICS system initialization table.
 - If you have a program list table defined and specified in your CICS system initialization table, add the HBRCCON program to the list by including the following statement:


```
DFHPLT TYPE=ENTRY,PROGRAM=HBRCCON
```
11. Optional: You can verify your configuration by running the Miniloan sample application as a CICS application. For more information see Sample: running the Miniloan application as a CICS application.

Configuring topology 4: Operational Decision Manager on WebSphere Application Server for z/OS











To use Rule Execution Server, the event runtime, or the Decision Center consoles on WebSphere Application Server for z/OS, you must deploy and configure the provided archives.

About this task

You deploy and configure the provided archives for WebSphere Application Server for z/OS in order to test and execute your applications in the environment that you use in production. You must first install the required products to get the necessary archives.

Use the following table to see which module is supported on each application server:

Table 3. Module support on WebSphere Application Server for z/OS

Application server	Rule Execution Server	Decision Center Enterprise console	Decision Center Business console	Event runtime	Event widgets
WebSphere Application Server for z/OS V8.5					
WebSphere Application Server for z/OS V8.0					

Configuring security on WebSphere Application Server for z/OS

Security configuration on WebSphere Application Server for z/OS can use the local operating system registry. Work with your z/OS system administrator to define the appropriate users, groups, and roles.

WebSphere Application Server for z/OS uses various kinds of user registries: local operating system, standalone Lightweight Directory Access Protocol (LDAP) registry, a standalone custom registry, and federated repositories. Only one type of user registry can be active. This active registry is shared by all product server processes. On Linux, WebSphere Application Server provides a custom registry sample named `FileRegistrySample`. On z/OS, WebSphere Application Server can use the local operating system registry. In this scenario, users, groups and roles are defined outside of WebSphere Application Server. Contact your z/OS system administrator to know which security registry and settings you require, and let them know which users and groups you need at the operating system level.

You must enable WebSphere Application Server administrative security and application security. You can deactivate the operating system check in the advanced properties of the Local operating system registry in WebSphere Application Server for z/OS. However, if you do deactivate this check, you must map the roles.

For more information, see [Configuring local operating system registries](#).

Configuring RACF security:

You can configure RACF security by using the JCL batch scripts provided.

Checking the existing setup:

You must confirm that WebSphere Application Server for z/OS uses the RACF local operating system security repository before proceeding. Security is configured when the application server is installed.

Before you begin

Make sure WebSphere Application Server for z/OS is installed to use RACF before you start to change settings through the Integrated Solutions Console.

About this task

When you install WebSphere Application Server for z/OS, you must set a number of options for storing security information. The two most common methods consist of using a federated repository or a local operating system repository, for example, RACF. When WebSphere Application Server for z/OS uses a RACF repository, security access for applications is managed in RACF through the z/OS system administrator (or whomever is in charge of RACF). For more information about RACF commands, refer to the z/OS V1R12.0 Security Server RACF Command Language Reference SA22-7687-15.

Note: With application security, Rule Execution Server applications use **EJBROLE** role values to determine which users and groups are allowed access and what tasks they can perform. These roles must be defined in RACF. See “Creating EJB roles and RACF groups” on page 36.

Procedure

1. In the side panel, open **Security** > **Global security**.
2. If **Local operating system** is already selected under **Current realm definition**, make sure **Enable application security** is selected.

If you select **Enable application security** you must click **Apply** and **Save** to apply the changes to the master configuration.

Note: Perform the following steps after your WebSphere Application Server for z/OS administrator has set up this WebSphere Application Server for z/OS instance to use the local operating system security provider.

- a. Click **Security Configuration Wizard**.
 - b. In **Step 1**, to specify the level of protection, select **Enable application security** and click **Next**.
 - c. In **Step 2**, select **Local operating system** and click **Next**.
 - d. In **Step 3**, set the name in the **Primary administrative user name** field and click **Next**.
 - e. In **Step 4**, review the security configuration summary and click **Finish**.
 - f. Click **Save** to save the changes to the master configuration.
 - g. Restart WebSphere Application Server for z/OS.
- You must log into the Integrated Solutions Console as the primary administrative user.
3. Check whether the HLQ property has been set.

If it is not set and you expect it to be, confirm the situation with your system administrator.

When you use RACF, you can set a high level qualifier (HLQ) at installation time to differentiate between separate installations of WebSphere Application Server for z/OS.

- a. Open **Security** > **Global security**, then click **Custom properties**.
- b. In the list of properties, check for the HLQ property named **com.ibm.security.SAF.profilePrefix** and make a note of its value. The SAF profile prefix is used for the ++EJBHLQ++ property. For more information on ++EJBHLQ++ see the HBRINST variables used to configure WebSphere Application Server for Operational Decision Manager for z/OS.

Related information:

“z/OS configuration variables” on page 117

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Creating EJB roles and RACF groups:

You create a set of EJB roles and RACF groups to give users various access rights to Decision Server for z/OS and Decision Center Decision Center.

About this task

Operational Decision Manager provides the predefined Enterprise JavaBeans (EJB) roles that are shown in the following table:

Table 4. EJB roles used to configure access rights

Module	EJB roles
Rule Execution Server	resAdministrators resDeployers resMonitors
Decision Center	rtsAdministrator rtsConfigManager rtsInstaller rtsUser
Decision Server Events	WBEAdmins WBEUsers WBEDashboardUsers WBERestApiUsers WBERestApiPrivilegedUsers WBEHTTPConnectorUsers HTTPEventConnectorUser EventWidgetsUsers

Operational Decision Manager includes JCL batch scripts that you run to create the necessary EJB roles. The scripts also create a RACF group for each EJB role. When you run the HBRUPTI job in “Step 5: Creating the working data sets” on page 14, it customizes the scripts with the values you specified in SHBRPARM(HBRINST). You can also manually customize the RACF group names by editing the scripts before you run them.

After you run the scripts, you assign users to the RACF groups associated with the roles you want to give them. For example, to authorize users to access Decision Center administrative functions, you add them to the RACF group associated with the rtsAdministrator EJB role.

The following ++HBRWORKDS++ SHBRJCL data set members contain the scripts. ++HBRWORKDS++ represents the high-level qualifier that is assigned to the working data sets.

- HBRDSRCF for Rule Execution Server
- HBREVRFCF for Decision Server Events
- HBRDCRCF for Decision Center.

Note: Run the scripts just for the Operational Decision Manager components you plan to use.

Procedure

1. Optional: Change the names of the RACF groups that are created by the scripts by editing the data set members.
2. Run the JCL batch scripts.
3. Review the log output in SDSF to ensure that the scripts ran correctly.
4. Use the RACF **CONNECT** command to assign users to the groups associated with their respective roles.

Related tasks:

Securing Decision Server Events

Configuring a federated repository:

Use the following steps if you configure either Rule Execution Server or Decision Center on WebSphere Application Server for z/OS and manage security through a federated repository. You map users to WebSphere Application Server roles to ensure the correct security levels.

You must carry out the following steps before you map the WebSphere Application Server roles:

1. Confirm that WebSphere Application Server for z/OS is configured for use with federated repositories. A federated repository is configured when the application server is installed.
2. Complete the configuration of Rule Execution Server and Decision Center by using the provided scripts or going through the manual steps.

Creating users and groups:

Create users and groups and assign them roles using a user registry with a federated repository.

Before you begin

If you do not have groups and users defined or if you want to define new groups and users, proceed with the following steps. If you already have suitable groups and users defined that can match the expected roles, skip this procedure and connect users and groups to their appropriate roles when you deploy your applications.

About this task

WebSphere Application Server uses various kinds of user registries (OS, LDAP, or Custom). You can configure a user registry with a federated repository.

The following variables in the `++HBRHLQ++`.SHBRPARG(HBRINST) member create the specified users and groups in the local repository, which map to the Rule Execution Server console and Decision Center in WebSphere Application Server.

Table 5. Variables for users and groups

Variable	Default value	Description
++RESADMIN++	resAdministrators	The administrator user group for the Rule Execution Server.
++RESDEPLOY++	resDeployers	The deployer user group for the Rule Execution Server.
++RESMONITOR++	resMonitors	The monitor user group for the Rule Execution Server.
++RESADMINUSER++	resAdmin	An admin user for the Rule Execution Server.
++RESDEPLOYUSER++	resDeployer	A deployer user for the Rule Execution Server.
++RESMONITORUSER++	resMonitor	A monitor user for the Rule Execution Server.
++RTSADMIN++	rtsAdministrator	The administrators user group for Decision Center.
++RTSCONFIG++	rtsConfigManager	The configuration user group for Decision Center.
++RTSUSER++	rtsUser	The users user group for Decision Center.
++RTSINSTALLER++	rtsInstaller	The installers user group for Decision Center.
++RTSADMINUSER++	rtsAdmin	An administrator for Decision Center.
++RTSCONFIGUSER++	rtsConfig	A configuration user for Decision Center.
++RTSUSERUSER++	rtsUser1	A user for Decision Center.

In most cases, the default values are sufficient, but if you want to use custom groups and add other users you can change these values. The user passwords are set to the same as the user id. You can add new users or edit these users in the WebSphere Application Server console under **Users and Groups**.

Any user of Decision Center must belong to at least one of the mandatory groups `rtsAdministrator`, `rtsConfigManager`, `rtsInstaller`, or `rtsUser`. Adherence to these groups determines the parts of Decision Center that a user can access. You must create all these groups in the application server. For testing purposes, also create a default user and password for each of these groups.

In addition, if you want to perform the Decision Center permissions tutorial in your own installation, you must create two custom groups: `Validator` and `Eligibility`.

Note:

- If you do not have groups and users defined or if you want define new groups and users, proceed through the steps below.
- If you already have groups and users defined that can match the expected role, you will connect them to the roles during the application deployment.

Procedure

To configure a user registry with a federated repository:

1. Log in to the Integrated Solutions Console.
2. In the side panel open **Security** > **Global security**.
3. Configure security as follows:
 - If **Federated repositories** is already selected under **Current realm definition**, make sure that **Enable application security** is selected. If you select **Enable application security**, you must click **Apply** and **Save** to save the changes to the master configuration.
 - If **Federated repositories** is not already selected, click **Security Configuration Wizard** and complete the wizard as follows:
 - a. In **Step 1**, to specify the level of protection, select **Enable application security** and click **Next**.
 - b. In **Step 2**, select **Federated repositories** and click **Next**.
 - c. In **Step 3**, type the name in the **Primary administrative user name** field and websphere in the **Password** field, and then click **Next**.
 - d. In **Step 4**, review the security configuration summary and click **Finish**.
 - e. Click **Save** to save the changes to the master configuration.
 - f. Restart WebSphere Application Server.

You must log into the Integrated Solutions Console as the primary administrative user. The secure port is used, for example
`https://example.hursley.ibm.com:1234/ibm/console/logon.jsp`
4. In the side panel open **Users and Groups** > **Manage Groups**, and click **Create**.
5. Enter **resAdministrators** as the group name, type a description if required, and click **Create**.
6. Click **Create Like** and create another group named **resDeployers**. Type a description if required. Click **Create**.
- 7.
8. Click **Create Like** again and enter another group named **resMonitors**. Type a description if required. Click **Create**. Click **Close**.
9. In the side panel open **Users and Groups** > **Manage Users**. Click **Create**.
10. Enter **resAdmin** as the User ID with a password **resAdmin**. Also specify the given name and last name.
11. Click **Group Membership**, then click **Search**. Select the **resAdministrators**, **resDeployers** and **resMonitors** groups and click **Add**. Click **Close**, then click **Create** and **Close** again.
12. Create another user named **resDeployer** with password **resDeployer**. Repeat the previous step and assign the user to the **resDeployers** and **resMonitors** groups.

Then create a user named **resMonitor** with password **resMonitor**. Repeat the previous step and assign the user to the **resMonitors** group.
13. Create the Decision Center administrator group.
 - a. Enter **rtsAdministrator** as the group name.
 - b. Type a description if appropriate.
 - c. Click **Create**.
14. Create the configuration manager group.

- a. Click **Create Like** and create another group named `rtsConfigManager`
 - b. Type a description if appropriate.
 - c. Click **Create**.
15. Create the installer group.
 - a. Click **Create Like** again and enter another group named `rtsInstaller`.
 - b. Type a description if appropriate.
 - c. Click **Create**.
 16. Create the users group. Type a description if required. Click **Create**.
 - a. Click **Create Like** again and enter another group named `rtsUser`
 - b. Type a description if appropriate.
 - c. Click **Create**.
 - d. Click **Close**.
 17. Create an administrator user.
 - a. In the side panel, open **Users and Groups** > **Manage Users** and click **Create**.
 - b. Enter `rtsAdmin` as the User ID and again `rtsAdmin` as a password , then the given name and last name.
 - c. Define group membership.
 - 1) Click **Group Membership**, then click **Search**.
 - 2) Select the `rtsAdministrator`, `rtsConfigManager`, `rtsInstaller`, and `rtsUser` groups and click **Add**
 - 3) Click **Close**, then click **Create**, and then **Close** again.
 18. Repeat 17 to create a configurator with `rtsConfig` as the user ID and again `rtsConfig` as the password.
 19. Repeat 17 again to create a configuration manager with `rtsConfigManager` as the user ID and password.
 20. Finally, repeat 17 again to create an installer with `rtsInstaller` as the user ID and password.
 21. Create a user with `rtsUser1` as the user ID and password, and then assign it to the `rtsUser` group.
 22. Restart your application server or your deployment manager.

Mapping the resAdministrators group to the Monitor role:

Map the `resAdministrators` users to the WebSphere Monitor role when you are using a federated repository so Rule Execution Server has sufficient security credentials to access MBeans.

About this task

To access the MBeans of the Rule Execution Server model, an application must have sufficient security credentials, restricted to the *Monitor* role in the WebSphere authentication system.

Rule Execution Server users can be given access to the MBeans of the model by configuring a mapping between the `resAdministrators` group declared in the custom registry and the *Monitor* role.

Procedure

1. In the Integrated Solutions Console open **Users and Groups** > **Administrative group roles**.
2. Click **Add**. For **Role(s)** select **Monitor**, then click **Search** and move the entry beginning with **resAdministrators** from the **Available** column to the **Mapped to role** column (use the arrow button).
3. Click **OK**.
4. Click the **Save** directly to the master configuration.
5. Restart your application server or your deployment manager.

Results

After you complete this step you can complete the configuration by following the steps in “Automated configuration of Rule Execution Server on WebSphere Application Server for z/OS,” or by using the Integrated Solutions Console to complete the steps starting with “Step 2: Configuring JDBC” on page 45.

Configuring Rule Execution Server on WebSphere Application Server for z/OS

After you set up security on the application server, you must then choose how you configure Rule Execution Server on a new instance of WebSphere Application Server for z/OS. You can do this task using `wsadmin` scripts or follow the manual steps.

Automated configuration of Rule Execution Server on WebSphere Application Server for z/OS:

One way to configure Rule Execution Server on a new instance of WebSphere Application Server for z/OS is to run the provided `wsadmin` scripts, which can be used to complete the deployment and configuration steps.

Before you start:

You can configure Rule Execution Server on WebSphere Application Server for z/OS with DB2 by running a set of scripts.

Important:

During the execution of the `wsadmin` scripts, the WebSphere Application Server might be restarted. If this is a problem, perform a manual configuration.

The process is as follows:

1. Complete the prerequisites (authorize the load library, customize the variables, decide on a topology, create the working data sets and directory, configure DB2, and configure security):
 - a. “Configuring,” on page 1
 - b. “Configuring security on WebSphere Application Server for z/OS” on page 34
2. Run the JCL job as indicated in “Submitting the HBRDSWAS JCL job” on page 42.
3. Deploy the MBean descriptors to enable the management beans, see “Deploying the Rule Execution Server MBean descriptors” on page 42.

Submitting the HBRDSWAS JCL job:

Submit the JCL job to complete the Rule Execution Server configuration.

About this task

After you have configured the JCL scripts in the ++HBRWORKDS++.SHBRJCL data set as explained in “Step 3: Customizing the z/OS configuration and runtime variables” on page 9, you submit the job to configure Rule Execution Server.

Procedure

1. Submit the ++HBRWORKDS++.SHBRJCL (HBRDSWAS) JCL job.

When the job is finished, you should see this message:

```
[exec]-----  
[exec] RES install tasks all done  
[exec]-----  
...  
BUILD SUCCESSFUL
```

2. Check the job spool **STDOUT** and **STDERR** for the logs.

If the job fails, the installation properties for WebSphere Application Server for z/OS can be found in the ++HBRWORKDS++.SHBRWASC (HBRRES) data set member, which is created in “Step 5: Creating the working data sets” on page 14 of the initial configuration.

Deploying the Rule Execution Server MBean descriptors:

To configure Rule Execution Server for WebSphere Application Server for z/OS, you must also deploy the MBean descriptors.

About this task

The Rule Execution Server architecture is based on the Java Management Extensions (JMX) API. MBeans are Java objects that are used by the JMX API. To configure Rule Execution Server for WebSphere Application Server, you must deploy the MBean descriptors, either globally for all Rule Execution Server instances or for a single Rule Execution Server instance.

To deploy to a targeted server instance, follow the procedure below.

Procedure

To deploy MBean descriptors:

1. Open the WebSphere Integrated Solutions Console.
2. In the side panel, open **Servers** > **Server Types** > **WebSphere application servers**.
3. On the Application servers page, click the name of your server.
4. Under Server Infrastructure, expand **Java and Process Management** and click **Process definition**.

In WebSphere Application Server for z/OS, an additional layer provides three resources that can be administered: Adjunct, Control, and Servant. If you are working in that environment, select the Servant.

5. Under Additional Properties, click **Java Virtual Machine**.
6. In the **Classpath** field, add <InstallDir>/executionserver/lib/jrules-mbean-descriptors.jar.

7. Click **OK**, then **Save** to save the changes directly to the master configuration.

Results

You have successfully configured Rule Execution Server on WebSphere Application Server for z/OS.

What to do next

You can now develop applications to execute rules using Rule Execution Server on WebSphere Application Server for z/OS. For more information, see Enabling applications to call Rule Execution Server on WebSphere Application Server for z/OS through WOLA and Developing client applications.

Manual configuration of Rule Execution Server on WebSphere Application Server for z/OS:

One way of configuring Rule Execution Server on a new instance of WebSphere Application Server for z/OS is to manually do the steps one by one.

Before you start:

Before you complete the configuration, review the software prerequisites and configuration steps for Rule Execution Server on WebSphere Application Server for z/OS.

Before you start a WebSphere Application Server for z/OS configuration, you must complete the initial configuration tasks that are described in “Configuring,” on page 1 (authorize the load library, customize the variables, decide on a topology, create the working data sets and directory, configure DB2, and configure security).

You can use wsadmin scripts to configure the WebSphere Application Server for z/OS as an alternative to “Automated configuration of Rule Execution Server on WebSphere Application Server for z/OS” on page 41.

Specific integration extension is available. For more information, see IBM® Support Pacs page.

Software Prerequisites

You must have the following software that is installed before you configure Rule Execution Server:









- WebSphere Application Server for z/OS
- DB2 for z/OS Version 9.1 or Version 10
- Rule Execution Server on WebSphere Application Server for z/OS.

You also need the following rights:

- Access to the WebSphere Application Server Integrated Solutions Console
- DB2 Administrator authority to CREATE a database, and CREATE and UPDATE tables
- Authority to start and stop the WebSphere Application Server
- Confirmation that WebSphere Application Server for z/OS is using the local operating system security repository (RACF) before proceeding. Security is configured at the time WebSphere Application Server is installed.

Configuration steps

The following table summarizes the steps to configure Rule Execution Server on WebSphere Application Server for z/OS.

Configuration Steps	Required
"Step 1: Configuring security on WebSphere Application Server for z/OS"	
You can now either complete the configuration by following the steps in "Automated configuration of Rule Execution Server on WebSphere Application Server for z/OS" on page 41, or by using the Integrated Solutions Console to complete the following configuration steps.	
"Step 2: Configuring JDBC" on page 45	
"Step 3: Creating the DB2 data sources and testing the connection" on page 46	
"Step 4: Setting the component and container managed aliases for the data source" on page 47	
"Step 5: Setting the currentSchema parameter" on page 47	
"Step 6: Deploying the Rule Execution Server MBean descriptors" on page 48	
"Step 7: Deploying the XU RAR" on page 48	
"Step 8: Deploying the Rule Execution Server Management EAR" on page 50	
"Step 9: Deploying the hosted transparent decision service EAR" on page 51	Recommended
"Step 10: Setting the web container custom property" on page 52	If you deploy the hosted transparent decision service EAR, you must complete this step.
Verifying your rules execution environment configuration	Recommended

Step 1: Configuring security on WebSphere Application Server for z/OS:

When you choose to configure Rule Execution Server on WebSphere Application Server for z/OS by using DB2, the first step consists in setting up administrative security and application security.

Rule Execution Server access is managed by the application server security. To access Rule Execution Server in WebSphere Application Server for z/OS, you must define a user registry as explained in "Configuring security on WebSphere Application Server for z/OS" on page 34. Two examples are provided, one using RACF, the other using a federated repository.

For information on WOLA security configuration:

- WebSphere Application Server 8.0: Securing optimized local adapters.

For information on WebSphere Application Server 8.5 see: WebSphere Application Server V8.5 information center.

Step 2: Configuring JDBC:

Create a JDBC provider in WebSphere Application Server for z/OS to enable Rule Execution Server.

Setting WebSphere Application Server for z/OS variables:

To create JDBC, you must first set WebSphere Application Server for z/OS variables.

Procedure

1. Navigate to **Environment > WebSphere variables** and click **DB2UNIVERSAL_JDBC_DRIVER_PATH**.
2. In the **Value** field, type the directory location for db2jcc.jar and db2jcc_license_cisuz.jar.
For example: /usr/lpp/db2910/classes
3. Click **Apply**, then click **Save**.
4. Click **DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH**.
5. In the **Value** field, type the native library path.
For example: /usr/lpp/db2910/lib
6. Click **Apply**, then click **Save**.

Creating the J2C authentication alias:

After you have set the variables for WebSphere Application Server for z/OS, you create the J2C authentication alias.

Procedure

1. Navigate to **Security > Global security** and expand **Java Authentication and Authorization Service**.
2. Click **J2C authentication data**, then click **New**.
3. Type resDB2user in the **Alias** field.
4. Type an active RACF user ID in the **User ID** field.
5. Type the password for the RACF User ID in the **Password** field.
6. Click **Apply**, then click **Save**.

Creating the JDBC provider:

After you set variables for WebSphere Application Server for z/OS and create the J2C authentication alias, you finish configuring JDBC by creating the JDBC provider.

About this task

You can install the JDBC provider at the cell, node, cluster, or server level. In Operational Decision Manager clustered environments, the JDBC provider is

typically configured at the node level. This procedure documents the installation at server level.

Procedure

1. Navigate to **Resources > JDBC > JDBC providers**.
2. In the **Scope** side panel, select **Node=xxx**, **Server=yyy**, where *xxx* is the name of your node and *yyy* the name of your server, and then click **New**.
3. In **Step 1**, select the following values:

Option	Description
Database type	DB2
Provider type	DB2 Universal JDBC Driver Provider
Implementation type	Select XA datasource , then type a name such as, for example, DB2 Universal JDBC Driver Provider for RES.

4. Click **Next**.
5. Skip **Step 2**.
A summary is provided in **Step 3**.
6. Check that the class path to the JAR file of your driver and the implementation class are correct. Default values are usually sufficient.
7. Click **Finish** and **Save** to save the changes to the master configuration, making sure that the **Synchronize changes with Nodes** check box is selected.

Step 3: Creating the DB2 data sources and testing the connection:

After you create your JDBC provider, create data sources to abstract the communication with the DB2 database away from the provider. You need data sources for storing RuleApps and Execution Object Models (XOMs).

About this task

If WebSphere Application Server is used in cluster mode, you must define the data sources at node level in the cluster not at the cluster level. For more information, see Rule Execution Server Deployment on cluster environments

Procedure

1. In the WebSphere Integrated Solutions Console, open **Resources > JDBC > Data sources**.
2. In the **Scope** side panel, select the scope that you set for the JDBC provider in "Step 2: Configuring JDBC" on page 45.
3. Click **New** to create a new data source.
4. Complete the following steps to create the data source for RuleApps:
 - a. In **Step 1**, enter a name for the data source in the **Data source name** field and `jdbc/<your_data_source_name>` in the **JNDI name** field, and click **Next**.
 - b. In **Step 2**, select the JDBC provider that you created in "Step 2: Configuring JDBC" on page 45 and click **Next**.
 - c. In **Step 3**, enter the following specific database properties for the data source:
 - Driver type:** 4.
 - Database name:** Your database name, for example DSN910GP.

Server name: Your DB2 server name.

Port number: Your DB2 port number.

- d. Clear the **Use this data source in container managed persistence (CMP)** option, then click **Next**.
 - e. In **Step 4**, set up any required security aliases and click **Next**. For more information, see “Step 4: Setting the component and container managed aliases for the data source.”
In **Step 5**, a summary of the data source is provided.
 - f. Click **Finish** and **Save** to save the changes to the master configuration.
5. Complete the following steps to increase the default connection pool size for the data source:
 - a. Edit the data source and click **Connection pool properties**.
 - b. Enter the value 30 into the **Maximum connections** field.
 - c. Click **OK**.

Note: The default value is 30 but you might require a larger **Maximum connections** value if your WebSphere Application Server is running many concurrent threads.

6. Create the data source for Java XOMs by repeating Step 4 on page 46 with the appropriate values. The default name of the XOM data source is jdbc/xomdatasource.
7. In the side panel, open **Resources** > **JDBC** and click **Data sources**.
8. Select the check box next to the data source that you want to test and click **Test connection**.

The status of the connection is indicated at the top.

Step 4: Setting the component and container managed aliases for the data source:

As part of setting up security, you set the authentication aliases, managed by the component and by the container for the resDB2user alias on resdatasource.

Procedure

1. Open **Resources** > **JDBC** and **Data sources** and click **resdatasource**.
2. In the **Security settings** section for **Component-managed authentication alias**, select the <node name>/resDB2user alias, where <node name> is the name of the WebSphere Application Server node on which you are configuring Rule Execution Server.
3. Select <node name>/resDB2user for the **Container-managed authentication alias**.
4. Click **Apply** and **Save** to save directly to the master configuration.

Step 5: Setting the currentSchema parameter:

Set the **currentSchema** parameter for resdatasource.

Procedure

1. Open **Resources** > **JDBC** and **Data sources** and click **resdatasource**.
2. Under **Additional Properties**, click **Custom properties**, then click **currentSchema** and type the name required in the **Value** field.
For a single server, the value is the schema name of your tables. For example: HBR1.
3. Click **Apply** and **Save** directly to the master configuration.

Step 6: Deploying the Rule Execution Server MBean descriptors:

To configure Rule Execution Server for WebSphere Application Server, you must also deploy the MBean descriptors.

Before you begin

Make sure that you give the application server process enough access right to read the `jrules-mbean-descriptors.jar` file. For example, change the permissions on the file by using a `chmod 777` command.

About this task

The Rule Execution Server architecture is based on the Java Management Extensions (JMX) API. MBeans are Java objects that are used by the JMX API. To configure Rule Execution Server for WebSphere Application Server, you must deploy the MBean descriptors, either globally for all Rule Execution Server instances or for a single Rule Execution Server instance.

Procedure

To deploy MBean descriptors:

1. Open the WebSphere Integrated Solutions Console.
2. In the side panel, open **Servers > Server Types > WebSphere application servers**.
3. On the Application servers page, click the name of your server.
4. Under Server Infrastructure, expand **Java and Process Management** and click **Process definition**.
In WebSphere Application Server for z/OS, an additional layer provides three resources that can be administered: Adjunct, Control, and Servant. If you are working in that environment, select the Servant.
5. Under Additional Properties, click **Java Virtual Machine**.
6. In the **Classpath** field, add `<InstallDir>/executionserver/lib/jrules-mbean-descriptors.jar`.
7. Click **OK**, then **Save** to save the changes directly to the master configuration.

Step 7: Deploying the XU RAR:

Deploy the execution unit (XU) RAR on WebSphere Application Server for z/OS.

About this task

This phase of the Rule Execution Server configuration consists in deploying the execution unit (XU) resource adapter archive (RAR) on WebSphere Application Server for z/OS and add the properties by which to identify it. The RAR file contains the XU and the persistence layer.

In the following procedure, `<version_number>` refers to the version of WebSphere Application Server for z/OS to which you deploy the XU RAR.

Important: In some cases, because of your application constraints, you might have to deploy the XU inside the application. It is your decision to choose the appropriate deployment mode of the XU: either embed it into the EAR or deploy it as a global connector. In any case, be aware of the following consequences.

- When the XU is deployed as a global connector:
 - The deployed Java EE applications might use its third-party libraries (such as ASM) instead of the libraries deployed in the application server.
 - Use a parent last setting for the XU Java EE application if your Java EE application does not support the version of the third-party libraries distributed with Decision Server. If you cannot use a parent last setting, you might need to embed the XU into the EAR that executes the rules.
- If you choose an embedded XU packaging, use a parent last setting for the code library if the third-party libraries version deployed at the level of the application-server code library are not compatible with the XU.

Procedure

1. Open the WebSphere Integrated Solutions Console.
2. In the panel, open **Resources** > **Resource Adapters** and click **Resource adapters**, then click **Install RAR**.
3. In the panel, select **Remote file system** and **Browse** to the location in UNIX System Services (USS) of z/OS where the archives reside:


```
<InstallDir>/executionserver/applicationservers/  
WebSphere<version_number>/jrules-res-xu-WAS<version_number>.rar
```
4. In **Scope**, select the node where you want to install the XU RAR file and click **Next**.
5. On the General Properties page, set the name for the XU, such as ILOG RES XU Resource Adapter.
6. If you use WebSphere Application Server v 8.5, check **Isolate this resource provider**.
7. Click **OK**.
8. Navigate to the created XU resource adapter and change the scope to view the adapter.
9. Under **Additional Properties**, click **J2C connection factories** and **New**.
10. Enter the following values:
 - **Name:** xu_cf
 - **JNDI name:** eis/XUConnectionFactory
11. Click **OK** and **Save** to save the changes to the master configuration.
12. Optional: If necessary, you can define more than one XU resource adapter.

Additional XU resource adapters are required when you have more than one node in your environment or you want to isolate the development environment and testing environment in one single node. On WebSphere Application Server for z/OS, you can install the XU resource adapter at the node level and then you can create another copy of the resource adapter that has a specific server or cluster as the scope.

For example, to define on the server level a XU resource adapter that has already been installed at the node level:

 - a. In the panel, open **Resources** > **Resource Adapters** and click **Resource adapters**.
 - b. Select the scope **Node=xxx**, **Server=yyy**, where *xxx* is the name of your node, *yyy* is the name of your server.
 - c. Click **New** and enter the name of the XU as XU.
 - d. In **Archive Path**, select `${CONNECTOR_INSTALL_ROOT}/jrules-res-xu-WAS<version_number>.rar`.

- e. In **Class Path**, enter `${CONNECTOR_INSTALL_ROOT}/jrules-res-xu-WAS<version_number>.rar`.
 - f. Click **OK**.
 - g. Repeat steps 5 on page 49 through 11 on page 49 to define the connection factory.
13. Restart the server.

Tip: Whenever you install or uninstall a XU, you must restart your application server.

What to do next

For more information, especially for instructions about packaging a connector into an EAR or about installing additional XU resource adapters, see the WebSphere Application Server Version 8.0 Information Center.

Step 8: Deploying the Rule Execution Server Management EAR:

Deploy the Rule Execution Server EAR file on WebSphere Application Server.

Procedure

1. Open the WebSphere Integrated Solutions Console.
2. Open **Applications > New Application** and click **New Enterprise Application**.
3. In the panel, select **Remote file system**, browse to the following file, and click **Next**.
 - For WebSphere Application Server V8.0: `<InstallDir>/executionserver/applicationservers/WebSphere8/jrules-res-management-WAS8.ear`
 - For WebSphere Application Server V8.5: `<InstallDir>/executionserver/applicationservers/WebSphere85/jrules-res-management-WAS85.ear`
4. Select the **Detailed - Show all installation options and parameters** check box.
5. Expand **Choose to generate default bindings and mappings**, select the **Generate Default Bindings** check box, and click **Next**.
6. Click **Continue** to accept the security warning.
7. In **Step 1**, click **Next** to accept the default settings.
8. In **Step 2**, proceed as follows:
 - If you have only one server, skip this step.
 - If you have more than one server, select the server to which you want to deploy the application, select the check box for **Rule Execution Server Console** and click **Next**.
9. In **Step 3** through **Step 8**, click **Next** to accept the default settings.

After **Step 6**, you receive a warning if you have not deployed a XU RAR yet. You can ignore this warning.
10. Depending on whether you use an RACF repository, proceed as follows:
 - If you use a RACF repository, ignore this step.
 - If you do not use a RACF repository, click **Step 9: Map security roles to users or groups** to map the security roles to users and groups:
 - a. Select the check box next to the **resAdministrators** role.
 - b. Click **Map groups** and click **Search**.

The groups are shown in a column titled **Available**.

- c. Select the **resAdministrators** item as follows:
 - 1) Click **resAdministrators** under **Available**.
 - 2) Click the arrow button to move **resAdministrators** to the **Selected** column.
- d. Click **OK** to return to the Mapping Users to Roles page.
- e. Repeat 10a through 10d to map the **resDeployers** and **resMonitors** roles for the other groups, making sure that only the check box next to the role that you are assigning is selected.

After you have completed the assignments, they are shown as follows:

Role	Mapped groups
resAdministrators	resAdministrators
resDeployers	resDeployers
resMonitors	resMonitors

- 11. In **Step 10** and **Step 11**, click **Next** to accept the default settings. **Step 12** provides a summary.
- 12. Click **Finish**.
- 13. When installation has completed, open **Application > Application Types > WebSphere enterprise applications**.
- 14. Click **ILOG Rule Execution Server**.
- 15. Click **Manage Modules**.
- 16. Click **Rule Execution Server Console**.
- 17. Under **General Properties**, for **Class loader order**, select **Classes loaded with local class loader first (parent last)**.
- 18. Click **OK**, then click **Save** to save directly to the master configuration.
- 19. In the panel, open **Applications > Application Types > WebSphere enterprise applications**.
- 20. In the Enterprise Applications page, select the check box next to **Rule Execution Servers** and click **Start** to start the application.

Step 9: Deploying the hosted transparent decision service EAR:

If you require the hosted transparent decision service, you must deploy the EAR file onto the same node as the execution unit (XU).

Procedure

- 1. Open the WebSphere Integrated Solutions Console.
- 2. In the pane, click **Applications > New Application and New Enterprise Application**.
- 3. In the panel, select **Remote file system**, browse to the archive files, and click **Next**.
`<InstallDir>/executionserver/applicationservers/
 WebSphere<version_number>/jrules-res-htds-WAS<version_number>.ear`
- 4. Select the check box **Detailed - Show all installation options and parameters**.
- 5. Expand **Choose to generate default bindings and mappings**, select the check box **Generate Default Bindings**, and click **Next**.
- 6. Click **Continue** to accept the security warning.
- 7. For **Step 1** through **Step 10**, click **Next** to accept the default settings.

Step 11 provides a summary.

8. Click **Finish**.
9. After the deployment has completed, click **Save** to save directly to the master configuration.
10. In the panel, open **Applications > Application Types > WebSphere enterprise applications** and click **jrules-res-htds-WAS8** or **jrules-res-htds-WAS7**.
11. Click **Manage Modules**.
12. Click **DecisionService**.
13. Under **General Properties**, for **Class loader order**, select **Classes loaded with parent class loader first**.
14. Click **OK**.
15. Click **OK**, then click **Save** to save directly to the master configuration.
16. In the panel, open **Applications > Application Types > WebSphere enterprise applications**.
17. In the Enterprise Applications page, select the check box next to **jrules-res-htds-WAS8** or **jrules-res-htds-WAS7** and click **Start** to start the application.
18. Set the **ruleset.xmlDocumentDriverPool.maxSize** ruleset property to the appropriate value. See Setting the ruleset.xmlDocumentDriverPool.maxSize property.

Step 10: Setting the web container custom property:

The hosted transparent decision service requires that you set the web container custom property `DecodeURLAsUTF8` to `False` to support a localized ruleset path.

About this task

This setting impacts all applications deployed on the server. For more information on checking that the hosted transparent decision service has been deployed successfully, refer to the Rule Execution Server console online help.

Procedure

To set the web container custom property:

1. Open the Websphere Integrated Solutions Console.
2. Click **Servers > Server Types > WebSphere application servers** and the server name.
3. Under **Container Settings**, click **Web container settings > Web container**.
4. Under **Additional Properties**, click **Custom properties**.
5. Click **New** and enter `DecodeURLAsUTF8` as the **Name** and `False` as the **Value**.
6. Click **Apply** and **Save** directly to the master configuration.

Results

You have now completed all of the steps to configure Rule Execution Server on WebSphere Application Server for z/OS. You can now run diagnostics to verify the configuration of Rule Execution Server. For more information, see Verifying the configuration.

What to do next

You can now develop applications to execute rules using Rule Execution Server on WebSphere Application Server for z/OS. For more information, see Enabling applications to call Rule Execution Server on WebSphere Application Server for z/OS through WOLA and Developing client applications.

Optional configuration steps:

After completing the steps to configure Rule Execution Server, you can enhance your configuration by adding support for WebSphere MQ for example, or by setting up a multiserver configuration.

Configuring Java execution units to connect to the zRule Execution Server console:

You can configure J2SE and J2EE execution units running on distributed platforms to connect to the zRules Execution Server console.

About this task

This task, and related tasks, are described in the distributed part of the Operational Decision Manager information center.

Configuring WebSphere Optimized Local Adapters (WOLA):

WebSphere Optimized Local Adapters (WOLA) is a feature of WebSphere Application Server for z/OS that manages communication between WebSphere Application Server and an external address space, such as CICS, batch, or IMS, that resides in the same logical partition.

You use WOLA with WebSphere Application Server for z/OS to perform bidirectional exchanges between your z/OS programs (COBOL or PL/I programs) and Rule Execution Server.

Important:

- You must install Rule Execution Server on WebSphere Application Server for z/OS before you configure WOLA.
- You must set the WOLA-related variables in the SHBRPARN(HBRINST) member before you configure your execution environment.

Configuring the WOLA load libraries on WebSphere Application Server for z/OS:

You use WebSphere Optimized Local Adapters (WOLA) in combination with WebSphere Application Server for z/OS to perform bidirectional exchanges between your z/OS applications (COBOL or PL/I applications) and Rule Execution Server. Install WOLA for ruleset execution from COBOL or PL/I.

Before you begin

Ensure that you have installed WebSphere Application Server for z/OS V8.0.0.3 or higher.

To use the WOLA feature, you must deploy an EJB application to an application server that runs in 64-bit mode. The EJB application uses the inbound connections of WOLA.

You must install WOLA on WebSphere Application Server if you want to execute rulesets from your z/OS application.

About this task

To configure the load libraries for WOLA, you must create symbolic links in the WebSphere Application Server configuration file system to the optimized local adapter modules and plug-in JAR files in the product file system. You must do this for each node that requires WOLA support.

Procedure

To install WOLA:

Follow the instructions in the WebSphere Application Server information center:

Option	Description
For WebSphere Application Server V8.0	Follow the instructions in Enabling the server environment to use optimized local adapters
For WebSphere Application Server V8.5	Follow the instructions in Enabling the server environment to use optimized local adapters

It is assumed that you install the WOLA load libraries in ++HBRWOLALOADLIB++. If you use IMS or CICS, ensure that the WOLA load module libraries are added to the STEPLIB list for IMS or CICS and that they are APF-authorized.

Deploying the WOLA EJBs:

To complete the configuration of WOLA, you deploy an EJB application that implements the WebSphere Optimized Local Adapters (WOLA) interface on WebSphere Application Server for z/OS v8.0, or v8.5.

About this task

The WOLA EAR file contains two EJB modules:

- `res-wola-proxy-ejb`: the entry point for the WOLA invocation. It accepts inbound requests from the API stub and delegates the requests to the worker EJB module.
- `res-wola-worker-ejb`: the EJB module that forwards the requests to Rule Execution Server.

The following procedure guides you through the installation of the WOLA EAR file on WebSphere Application Server for z/OS.

Procedure

1. In the WebSphere Application Server console, select **New Application > New Enterprise Application**.
2. Specify the path of the EAR file as `<InstallDir>/executionserver/applicationservers/WOLA/jrules-res-wola.ear`. Click **Next**.
3. In the "Preparing for the application installation" pane, perform the following actions:
 - a. Select **Detailed - Show all installation options and parameters**.
 - b. Expand **Choose to generate default bindings and mappings**.

- c. Select the check box **Generate Default Bindings** and click **Next**.
- 4. In **Step 1**, click **Next** to accept the default settings.
- 5. In **Step 2**, take one of the following actions:
 - a. If you have only one server, skip this step.
 - b. If you have more than one server, select the server to which you want to deploy the application, select the check box for the two modules, and click **Next**.
- 6. In **Step 3**, click **Next** to accept the default settings.
- 7. In **Step 4**, click **Next** to accept the default settings.
- 8. In **Step 5**, provide the following JNDI names for the beans:
 - a. For the EJB module `res-wola-proxy-ejb-<version>.jar`: select the option **JNDI name for all interfaces Target Resource JNDI Name** and enter the name `ejb/com/ibm/rules/wola/ProxyExecutionSessionBean`.
 - b. For the EJB module `res-wola-worker-ejb-<version>.jar`: select the option **JNDI name for all interfaces Target Resource JNDI Name** and enter the name `ejb/com/ibm/rules/wola/PojoExecutionSessionBean`.
 - c. Click **Next**.

Important: The JNDI names of the modules must be distinct, otherwise you can get a `NameAlreadyBoundException` when you start the EJB.

- 9. In **Step 6**, click **Next**.
- 10. In **Step 7** on WebSphere Application Server V8.0 and V8.5, map the EJB reference for the proxy bean to the worker bean by entering `ejb/com/ibm/rules/wola/PojoExecutionSessionBean` in the **Target Resource JNDI Name** column for `res-wola-proxy-ejb-<version>.jar`. Click **Next**.
- 11. In **Step 8**, complete the following actions:

Option	Description
WebSphere Application Server V8.0	Click Next until Finish
WebSphere Application Server V8.5	<ol style="list-style-type: none"> 1. For the <code>res-wola-worker-ejb</code> resource reference, <code>eis/XUConnectionFactory</code>, click Browse. 2. Select <code>eis/XUConnectionFactory</code> as the Target Resource JNDI Name. 3. For the <code>res-wola-worker-ejb</code> resource reference, <code>jdbc/resdatasource</code>, click Browse. 4. Select <code>jdbc/resdatasource</code> as the Target Resource JNDI Name. 5. Click Next. <p>If you cannot see these options when you click Browse, ensure that you have completed all the steps for installing Rule Execution Server on WebSphere Application Server for z/OS.</p>

- 12. In **Step 9**, click **Next**.
- 13. In **Step 10**, click **Next**.
- 14. In **Step 11**, click **Finish**.
- 15. After the EJB deployment is complete, click **Save** to save it to the WebSphere Application Server master configuration.

16. Start the application Decision Server WOLA EndPoint:
 - a. In the WebSphere Application Server console, select **Applications > Application Types > WebSphere enterprise applications**.
 - b. In the Enterprise Applications page, select the check box **Decision Server WOLA EndPoint**.
 - c. Click **Start**.

Configuring batch jobs to execute rules through WOLA:

To call Rule Execution Server on WebSphere Application Server for z/OS from your z/OS application (COBOL or PL/I application), you must specify the WebSphere Optimized Local Adapters (WOLA) load library, and the parameters required to locate the instance of WebSphere Application Server for z/OS.

About this task

To enable z/OS applications to call Rule Execution Server through WOLA, you first configure Rule Execution Server and WOLA on the same instance of WebSphere Application Server for z/OS. Then you modify the JCL that launches your z/OS application, and set the SHBRPARM(HBRWOLA) data set member with the appropriate values.

Procedure

1. Add the following line to the STEPLIB in the JCL that launches the application:
DD DISP=SHR,DSN=++HBRWOLALOADLIB++where ++HBRWOLALOADLIB++ is the location where you installed the WOLA load library.
2. To identify the parameter file, add the following line in the JCL that launches the application:
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRWOLA)
3. Modify the following lines in the parameter file ++HBRWORKDS++.SHBRPARM(HBRWOLA):
 - HBRTARGETRES=++HBRTARGETRES++
 - HBRWOLACELL=++HBRWOLACELL++
 - HBRWOLANODE=++HBRWOLANODE++
 - HBRWOLASERVER=++HBRWOLASERVER++

The last three lines must be modified to specify the WebSphere Application Server instance that you want to use.

If you are calling your application from IMS, add a DFSESL DD statement to your JCL. This statement must identify the WOLA load library and the IMS SDFSRESL library.

Configuring CICS to run rules through WOLA:

After you configure a Rule Execution Server on WebSphere Application Server for z/OS instance, you can configure CICS to run rules on the newly configured server through WOLA.

About this task

Use the following procedure to configure CICS to run rules on Rule Execution Server on WebSphere Application Server for z/OS through WOLA.

You can optionally configure CICS to run the HBRC transaction at startup to connect to the server instance automatically, if it is running. With this option, CICS is immediately ready to run rules. If the server is not running, CICS starts normally but you must run the HBRC transaction manually before rule execution can begin.

Procedure

1. Submit the `++HBRWORKDS++.SHBRJCL(HBRCSO)` job to define the resources that are necessary for CICS.
2. Submit the `++HBRWORKDS++.SHBRJCL(HBRWOLA)` job to define the resources that are necessary for WOLA under CICS.
3. In the CICS system initialization table, add the list names `BBOLIST` and `++CICSLIST++` to the list of resource definition groups that are specified by the `GRPLIST` parameter.

4. Edit the CICS JCL to include the Rule Execution Server data sets:

- a. Add the `++HBRHLQ++.SHBRICS` and the `++HBRWOLALOADLIB++` libraries to the `DFHRPL` concatenation.

- b. Pass the necessary runtime variables to the server by adding the `SHBRPAM(HBRWOLA)` and `SHBRPAM(HBRMMN)` data set members to the `HBRENVPR` DD statement. These variables are shown in the following DD statements:

```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPAM(HBRWOLA)
//          DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPAM(HBRMMN)
```

- c. If you plan to run the Miniloan sample application to verify your configuration, add the `SHBRPAM(HBRSCEN)` data set member as a `SCENARIO` DD statement. The `HBRSCEN` member contains input values to the Miniloan application.

```
//SCENARIO DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPAM(HBRSCEN)
```

5. Start the Rule Execution Server.
6. Start CICS and install the `HBRGROUP` resources by using the following command:

```
CEDA INSTALL GROUP(HBRGROUP)
```

7. Install the `BBOACSD` group by using the following command:

```
CEDA INSTALL GROUP(BBOACSD)
```

8. Activate the optimized local adapters `TRUE` program by using the following command:

```
BBOC START_TRUE
```

You can use the `BBOC` transaction to start, stop, enable, or disable tracing for the `TRUE` module.

9. Test the configuration by running the `HBRC` transaction to connect CICS to Rule Execution Server. If CICS succeeds in connecting to the server, message `GBRZC9001I` is produced. If CICS fails to connect to the server, message `GBRZC9000E` is produced with a return code that contains diagnostic information. If you forget to do `BBOC START_TRUE`, the message Transaction `HBRC` failed with abend `AEY9` is produced when `HBRC` is run.
10. Optional: To configure CICS to connect automatically at startup to the running server instance, take either of the following actions:
 - If you do not have a program list table that is defined, add the `PLTPI=HB` parameter to the CICS system initialization table.

- If you already have a program list table that is defined and specified in your CICS system initialization table, add the **HBRCCON** program to the list by including the following statement: `DFHPLT TYPE=ENTRY,PROGRAM=HBRCCON`.
11. Optional: To perform WOLA registration externally to Operational Decision Manager for z/OS, use the JCL variable `HBRWOLAREGNAME` to inform Operational Decision Manager for z/OS of the registration name to use. When the JCL variable `HBRWOLAREGNAME` is not set, the default is to call `BBOA1REG` when **HBRC** is called by using a randomly created name. For example, you might run the transaction `BBOC REG RGN=ILKREGNAME DGN=CILK NDN=NILK SVN=WSVRILK`, then set `HBRWOLAREGNAME=ILKREGNAME`.

Configuring IMS to execute rules through WOLA:

You can configure IMS to execute rules on Rule Execution Server on WebSphere Application Server for z/OS through WOLA.

Before you begin

Note the following requirements:

- Rule Execution Server on WebSphere Application Server for z/OS supports IMS versions 11 and 12.
- IMS applications can run in a message processing region (MPR) or a batch processing region (BMP). Applications that use DL/I calls are not supported.
- IMS and Decision Server for z/OS must run in the same logical partition (LPAR).

About this task

To enable z/OS applications (COBOL or PL/I applications) to call Rule Execution Server through WOLA, you configure Rule Execution Server and WOLA on the same instance of WebSphere Application Server for z/OS. Then you modify the IMS startup data, and initialize the `SHBRPARM(HBRWOLA)` data set member. If your applications run in a message processing region, you must also update the message processing region JCL.

For more information about the IMS-specific installation and setup requirements, see *Enabling optimized local adapters support in IMS*.

Procedure

1. Create an external subsystem IMS PROCLIB member, or update your existing member to include the following entry: `WOLA,BBOA,BBOAIEMT`.
2. Pass the SSM parameter into your IMS startup data.
3. Include the WOLA load library in the STEPLIB and the DFSESL DD statements in your IMS control region startup JCL.
4. If your IMS applications run in a message processing region, make the following changes to the message processing region JCL:
 - a. Add an `HBRENVPR` DD statement that specifies the zRule Execution Server for z/OS instance to execute rules. For example:


```
//HBRENVPR DD DISP=SHR,
// DSN=WODM.HBRR.CUSTOMISE.HBRR.SHBRPARM(HBRWOLA)
```
 - b. Add the load library that contains your IMS application to the STEPLIB and DFSESL DD statements. For example:


```
// DD DSN=WODM.HBRR.BASE.SHBRLOAD,DISP=SHR
```

Note: For optimum performance, consider putting the load library at the top of the STEPLIB or in the pageable link pack area (PLPA). For more information, see the Choosing IMS options for performance section in the IMS information center.

5. Use HBRCON and HBRDISC to connect and disconnect at rule execution time; this is necessary because the pre-initialization routine used for connecting to a stand-alone Rule Execution Server cannot be used with WOLA.
6. Restart IMS to register the changes.

Verifying the configuration:

Test your configuration by running the WOLA Miniloan sample on Rule Execution Server for z/OS .

About this task

After you configure the WOLA (WebSphere Optimized Local Adapters) load libraries and deploy the WOLA EJB EAR file, you need to make sure that the configuration works correctly. The simplest way to validate the functionality of WOLA is to use the Miniloan sample program. The sample runs in a batch job and calls the EJB application running on the server through WOLA.

Procedure

Run the Miniloan sample as a batch job using the Rule Execution Server and WOLA instance you have configured on WebSphere Application Server for z/OS. For more information, see Sample: Running the Miniloan application as a batch job

Integrating WebSphere MQ in WebSphere Application Server to support asynchronous execution:

If you use the WebSphere MQ messaging provider support in WebSphere Application Server, you can deploy Java EE applications that directly use the enterprise messaging capabilities of WebSphere MQ.

WebSphere MQ integration overview:

The Java Message Service (JMS) API enables access to rule services. To use a message-driven rule bean, you must create the necessary resources under the WebSphere MQ JMS provider. Both the publish-and-subscribe and the point-to-point models are supported.

A Decision Server rule service can be accessed by an asynchronous invocation pattern that uses the Java Message Service (JMS) API. When a JMS message arrives, the EJB container calls a message-driven rule bean (MDB). The MDB can reside locally or remotely from the client application. In turn, the message-driven rule bean calls the rulesets that are running in the execution unit (XU). The real call to the rule engine is delegated to a simple rule session.

In WebSphere Application Server, the client application is the scenario running in WebSphere Application Server that calls the rule service, the server is the application server where Rule Execution Server is installed. Rule Execution Server is usually running remotely to the client application.

To use a Decision Server message-driven rule bean, you must create the necessary resources under the WebSphere MQ JMS provider at the proper scope for both the

client and the server side to make them visible for the client application and the Decision Server MDB respectively. Decision Server implements both standards of asynchronous messaging: the Publish-and-Subscribe Model and the Point-to-Point Model. The following procedure demonstrates how to set up both a point-to-point messaging model and a publish-and-subscribe model. If you need only one of them, you can comment out the resource reference in the deployment descriptor of Decision Server MDB.

Before installation, you must create the following resources in WebSphere MQ:

- Queue: JRulesIn, JRulesOut
- Topic: JRulesTopicIn, JRulesTopicOut

Use these resources to submit rule execution requests and obtain execution results.

You must perform the following tasks to integrate WebSphere MQ in WebSphere Application Server:

Creating the WebSphere MQ queue connection factory:

To create the WebSphere MQ queue connection factory, you create the queue, then you configure the connection factory by following the connection factory wizard.

Procedure

To create a queue connection factory to connect WebSphere MQ with the Decision Server MDB:

1. Log in to the Integrated Solutions Console.
2. Open **Resources** > **JMS** and click **JMS Providers**.
3. In the panel, select the **WebSphere MQ messaging provider**.
4. Under **Additional Properties**, click **Queue connection factories**.
5. Click **New**.
6. In **Step 1: Configure basic attributes**, set the fields **Name** and **JNDI name** as follows, then Click **Next**.
Name JRules Queue Connection Factory
JNDI name
jms/BRESQueueConnectionFactory
7. In **Step 2: Select connection method**, select **Enter all the required information into this wizard** and click **Next**.
8. In **Step 2.1: Supply queue connection details**, type the name of your queue manager or queue sharing group, then click **Next**.
9. In **Step 2.2: Enter connection details**, type the connection details to establish a connection to the queue manager or queue sharing group, then click **Next**.
The default queue port is 1414.
10. In **Step 3: Test connection**, click **Test connection**.
If your message queue is running, you see the following message:
A connection was successfully made to WebSphere MQ.
11. Click **Next**.
A summary opens showing the details of the connection factory.
12. Click **Finish** and then click **Save** to save directly to the master configuration.

Creating the WebSphere MQ input queue:

After you have created the WebSphere MQ queue connection factory, you can now create the JMS queue destination for receiving a request message. To do so, you set the scope to node or server level, you select the provider, name the queue, and finally save the configuration.

Procedure

To create the JMS queue:

1. In the Integrated Solutions Console, open **Resources** > **JMS** and click **Queues**.
2. Set the scope to either Node level or Server level, then click **New**.
3. In the panel, select **WebSphere MQ messaging provider**, then click **OK**.
4. In General Properties, set the fields **Name**, **JNDI name**, and **Queue name** as follows:

Name JRules Input Queue

JNDI name
jms/BRESQueueIn

Queue name
JRulesIn

5. Type in the name of your queue manager or queue sharing group name, then click **OK**.
6. Click **Save** to save directly to the master configuration.

Creating the WebSphere MQ output queue:

After you have created the input queue for request messages, you must also create the JMS queue destination for sending a response message. To do so, you set the scope to node or server level, select the provider, name the queue, and save.

Procedure

To create the output queue:

1. In the Integrated Solutions Console, open **Resources** > **JMS** and click **Queues**.
2. Set the scope to either Node level or Server level, then click **New**.
3. In the panel, select **WebSphere MQ messaging provider**, then click **OK**.
4. In General Properties, set the fields **Name**, **JNDI name**, and **Queue name** as follows:

Name JRules Output Queue

JNDI name
jms/BRESQueueOut

Queue name
JRulesOut

5. Type in the name of your queue manager or queue sharing group name, then click **OK**.
6. Click **Save** to save directly to the master configuration.

Creating a topic connection factory:

After you have created the queue factory, input queue, and output queue, you must create a topic connection factory. To do so, you set the scope to node or server, select the provider and JNDI name, enter the connection details, test the connection, and save.

Procedure

1. In the Integrated Solutions Console, open **Resources** > **JMS** and click **Topic connection factories**.
2. Set the scope to either Node level or Server level, then click **New**.
3. In the panel, select **WebSphere MQ messaging provider**, then click **OK**.
4. In **Step 1: Configure basic attributes** set the fields **Name**, and **JNDI name** as follows and click **Next**.

Option	Description
Name	JRules Topic Connection Factory
JNDI name	jms/BRESTopicConnectionFactory

5. In **Step 2: Select connection method**, select **Enter all the required information into this wizard** and click **Next**.
6. In **Step 2.1: Supply queue connection details**, type the name of your queue manager or queue-sharing group, then click **Next**.
7. In **Step 2.2: Enter connection details**, type the connection details to establish a connection to the queue manager or queue sharing group (the default queue port is 1414), then click **Next**.
8. In **Step 3: Test connection**, click **Test connection**.
If your message queue is running, you see the following message: A connection was successfully made to WebSphere MQ.
9. Click **Next**. A summary opens showing the details of the connection factory.
10. Click **Finish**, then click **Save** to save directly to the master configuration.

Creating the WebSphere MQ input topic:

After you have created the WebSphere MQ topic connection factory, you can create the JMS topic destination for receiving a request message. To do so, you set the scope to node or server level, select the provider, set the JNDI and input topic names, and save.

Procedure

1. In the Integrated Solutions Console, open **Resources** > **JMS** and click **Topics**.
2. Set the scope to either Node level or Server level, then click **New**.
3. In the panel, select **WebSphere MQ messaging provider**, then click **OK**.
4. In General Properties, set the fields **Name**, **JNDI name**, and **Topic name** as follows, then click **OK**.

Option	Description
Name	JRules Input Topic
JNDI name	jms/BRESTopicIn
Topic name	JRulesTopicIn

5. Click **Save** to save directly to the master configuration.

Creating the WebSphere MQ output topic:

After you have created the WebSphere MQ topic connection factory and input topic, you must also create the JMS queue destination for sending a response message. To do so, you set the scope to node or server level, select the provider, set the JNDI and output topic names, and save.

Procedure

To create the JMS output topic:

1. In the Integrated Solutions Console, open **Resources** > **JMS** and click **Topics**.
2. Set the scope to either Node level or Server level, then click **New**.
3. In the panel, select **WebSphere MQ messaging provider**, then click **OK**.
4. In General Properties, set the fields **Name**, **JNDI name**, and **Topic name** as follows and then click **OK**.

Name

JRules Output Topic

JNDI name

jms/BRESTopicOut

Topic name

JRulesTopicOut

5. Click **Save** to save directly to the master configuration.

Creating the WebSphere MQ queue activation specification:

After you have configured WebSphere MQ queues, you create the queue activation specification.

About this task

The queue activation specification manages the relationship between the Decision Server message-driven rule beans (MDB) running in WebSphere Application Server and a destination in WebSphere MQ. To create the activation specification, you set the scope to node or server level, select the provider, set the specification and JNDI names, enter the connection details, and save.

Procedure

To create the activation specification:

1. In the Integrated Solutions Console, open **Resources** > **JMS** and click **Activation specifications**.
2. Set the scope to either Node level or Server level, then click **New**.
3. In the panel, select **WebSphere MQ messaging provider**, then click **OK**.
4. In **Step 1: Configure basic attributes** set the fields **Name**, and **JNDI name** as follows and then click **Next**.

Name

JRules Activation Spec

JNDI name

eis/IlrRuleExecutionEJB

5. In **Step 1.1: Specify MDB destination data**, set the field **Destination JNDI name** to `jms/BRESQueueIn`, set the Destination type to **Queue**, then click **Next**.
6. In **Step 2: Select connection method**, select **Enter all the required information into this wizard** and click **Next**.
7. In **Step 2.1: Supply queue connection details**, type the name of your queue manager or queue sharing group, then click **Next**.
8. In **Step 2.2: Enter connection details**, type the connection details to establish a connection to the queue manager or queue sharing group, then click **Next**.
The default queue port is 1414.
9. In **Step 3: Test connection**, click **Test connection**.
If your message queue is running, you see the following message:
A connection was successfully made to WebSphere MQ.
10. Click **Next**.
A summary opens showing the details of the connection factory.
11. Click **Finish**, then click **Save** to save directly to the master configuration.

Creating the WebSphere MQ topic activation specification:

After you have created the queue activation specification, you create the topic activation specification.

About this task

The topic activation specification manages the relationship between the Decision Server message-driven rule beans (MDB) running in WebSphere Application Server and a destination in WebSphere MQ. To create the activation specification, you set the scope to node or server level, select the provider, set the specification and JNDI names, enter the connection details, and save.

Procedure

To create the topic activation specification:

1. In the Integrated Solutions Console, open **Resources** > **JMS** and click **Activation specifications**.
2. Set the scope to either Node level or Server level, and then click **New**.
3. In the panel, select **WebSphere MQ messaging provider**, and then click **OK**.
4. In **Step 1: Configure basic attributes** set the fields **Name**, and **JNDI name** as follows, and then click **Next**.

Name JRules Topic Activation Spec

JNDI name

eis/IlrRuleExecutionTopicEJB

5. In **Step 1.1: Specify MDB destination data** set the field **Destination JNDI name** to `jms/BRESTopicIn`, set the Destination type to **Topic**, and then click **Next**.
6. In **Step 1.2: Configure Durable Subscription**, select **Nondurable subscription**, and then click **Next**.
7. In **Step 2: Select connection method**, select **Enter all the required information into this wizard** and click **Next**.

8. In **Step 2.1: Supply queue connection details**, type the name of your queue manager or queue sharing group, and then click **Next**.
The default queue port is 1414.
9. In **Step 2.2: Enter connection details**, type the connection details to establish a connection to the queue manager or queue sharing group, and then click **Next**.
10. In **Step 3: Test connection**, click **Test connection**.
If your message queue is running, you see the following message:
A connection was successfully made to WebSphere MQ.
11. Click **Next**.
A summary opens showing the details of the connection factory.
12. Click **Finish**, and then click **Save** to save directly to the master configuration.

Installing the message-driven rule bean:

After you have created the queue and topic activation specifications, you install the Decision Server message-driven rule bean (MDB) in WebSphere Application Server as an enterprise application. To do so, you create a new enterprise application.

Procedure

To install the Decision Server message-driven rule bean:

1. Open the Integrated Solutions Console.
2. In the panel, open **Applications > New Application** and click **New Enterprise Application**.
3. In the panel, select **Local file system** and **Browse** to the following path:
<InstallDir>/executionserver/applicationservers/
WebSphere<version_number>/jrules-res-mdb-WAS<version_number>.jar.
4. Click **Next**.
5. Select the check box **Detailed - Show all installation options and parameters**.
6. Expand **Choose to generate default bindings and mappings** and select the check box **Generate Default Bindings**.
7. Click **Next**, then click **Continue** to accept the security warning.
8. Click **Step 5: Bind listeners for message-driven beans**.
 - a. Type `jms/BRESTopicIn` as the Destination JNDI name for `IlrRuleExecutionTopicEJB`.
 - b. Type `jms/BRESQueueIn` as the Destination JNDI name for `IlrRuleExecutionEJB`.
 - c. Click **Next**.
9. Click **Step 6: Map resource references to resources** and then click **Next**.
Use the default binding for the referenced resources.
Step 7 provides a summary.
10. Click **Finish**.
11. When the installation has completed, click **Save** directly to the master configuration.
12. In the Integrated Solutions Console, open **Applications > Application Types > WebSphere enterprise applications**.

13. In the Enterprise Applications page, select the check box next to **jrules-res-mdb-WAS<version_number>.jar** and click **Start** to start the application.

Enabling server-wide Last Participant Support:

To complete the integration of WebSphere MQ, you enable Last Participant Support.

About this task

To finish integrating WebSphere MQ in WebSphere Application Server for asynchronous execution, you enable Last Participant Support (LPS) so that a single one-phase commit resource is used with any number of two-phase commit resources in the same global transaction. To do so, you set the `ACCEPT_HEURISTIC_HAZARD` custom property to true in the Integrated Solutions Console and then restart the application server.

Procedure

To enable server-wide LPS:

1. Open the **Integrated Solutions Console**.
2. Click **Servers** > **Server Types** > **WebSphere application servers** and the server name.
The properties page for the application server opens
3. Under **Container Settings**, expand **Container Services** and click **Transaction Service**.
The properties page for the transaction service opens.
4. Under **Additional Properties**, click **Custom properties**.
5. On the Custom Properties page, click **New** and type `ACCEPT_HEURISTIC_HAZARD` as the **Name** and `TRUE` as the **Value**.
6. Click **Apply** and **Save** directly to the master configuration.
7. Restart your application server.

Configuring Rule Execution Server in different environments:

General configuration guidelines apply if you want to enable Rule Execution Server on different environments in a single cell.

Before you begin

These instructions assume that you do not configure multiple instances of Rule Execution Server in a single cell.

About this task

Most likely, the development of your business rule management system (BRMS) requires more than a single deployment of Rule Execution Server. The development lifecycle of a business rule application is similar to any other software development process: It includes stages for implementation, testing, deployment, and maintenance. At the very least, you are likely to need an environment for your development team, one for your QA team, and another one for in-production applications. In the cases where you configure Rule Execution Server in a single cell, it is good practice to isolate the rulesets that you use on each server and

ensure that the execution units (XUs) do not interfere with each other.

Procedure

1. Set up different data sources.
Use unique JNDI names. For example: jdbc/resdatasourceEnv1 and jdbc/resdatasourceEnv2
2. Deploy a XU for each environment and define a J2C connection factory.
 - a. In the side panel, open **Resources > Resource Adapters > J2C connection factories** and click the name of the connection factory `xu_cf`. For more information about creating a connection factory, see Step 7: Deploying the XU RAR .
 - b. Modify the JNDI name to `eis/XUConnectionFactoryEnv1`.

Remember: The predefined JNDI name is `eis/XUConnectionFactory`. When you choose a different one, you must modify the execution components that call this XU so that they use this JNDI instead of the predefined one.
 - c. Under Additional Properties, click **Custom properties**.
 - d. Click the **plugins** property.
 - e. In the **Value** field, change `xuName=default` in the property to `xuName=xuEnv1`, and then click **OK**.
 - f. Click the **persistenceProperties** property.
 - g. In the **Value** field, change `JNDI_NAME=jdbc/resdatasource` to `JNDI_NAME=jdbc/resdatasourceEnv1`.
 - h. Click **OK** and **Save** to save the changes to the master configuration.
 - i. Repeat the entire process for XUs in other environments.
3. Deploy the Rule Execution Server console for each environment.
 - a. To modify the deployment descriptor of the Rule Execution Server console EAR file: in the `web.xml` file, uncomment the **JMX_XU_QUERY_PART** parameter and specify `xuName=xuEnv1`.
 - b. Deploy the Rule Execution Server console EAR on the server in the resource reference settings in the application server.
 - 1) Set the JNDI for the data source to `jdbc/resdatasourceEnv1`.
 - 2) Set the JNDI name for the XU to `eis/ConnectionFactoryEnv1`.
 - c. Repeat the process to deploy the Rule Execution Server console for the other environments.
4. Restart the node agents after you complete the configuration.
5. Call the XU instances to register the XU with the Rule Execution Server console.

Rule Execution Server deployment on cluster environments:

Taking into account cluster configuration and topology, you can deploy Rule Execution Server to WebSphere Application Server cell in cluster mode.

Cluster configuration and topology:

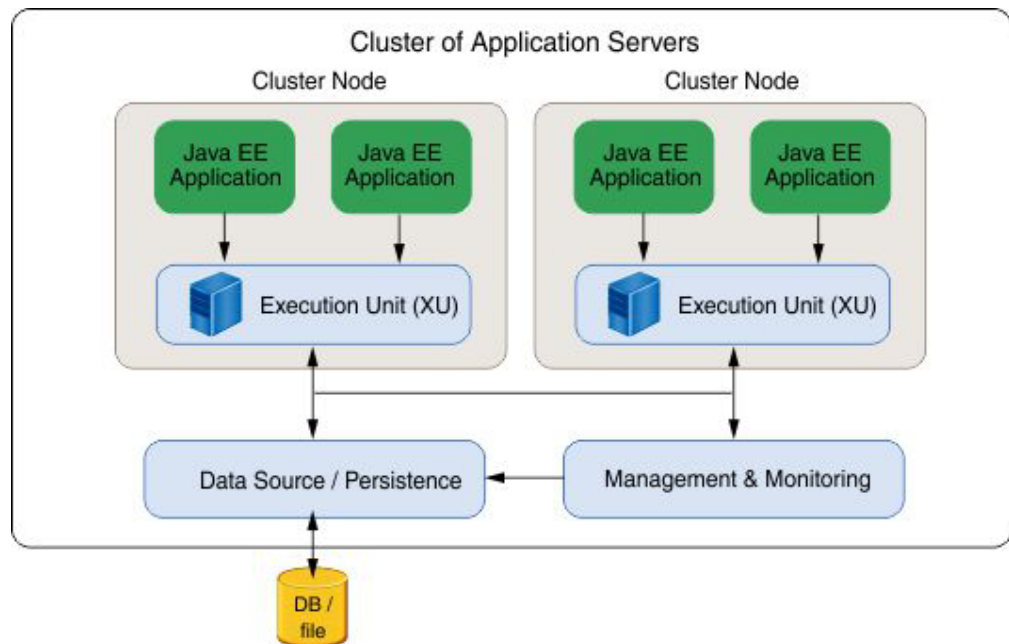
When you deploy Rule Execution Server onto server clusters, each node hosts one execution unit (XU), which is used only by a local rule session. The cluster topology significantly affects the notification mechanism.

Within the Java EE framework, clusters provide mission-critical services to ensure minimal downtime and maximum scalability. A cluster is a group of application servers that transparently run your Java EE application as if it were a single entity.

Cluster implementations on Java EE application servers come with their own set of terminology. Each of the following terms are important to understand how your cluster performs:

- Cluster and component failover services
- HTTP session failover
- Single points of failure in a cluster topology

On a cluster configuration, deploy an execution unit (XU) on each node. There is one XU for each node of a cluster. Use the administration console of application servers to handle cluster deployment. A XU instance can be used only by a local (same node) rule session. The rule session and the XU communicate through direct Java method calls, so the XU does not require serialization.



A cluster that uses Rule Execution Server involves a collaboration between the Rule Execution Server MBeans. The topology of the cluster has significant influence on the management of the notification mechanism when a resource is changed.

The management model is likely to use several times a basic scenario of a distributed notification mechanism within a cluster to interact with the various execution unit (XU) instances. A XU message-driven rule bean (MBean) is deployed with the XU to collaborate with the Rule Execution Server JMX infrastructure.

The following sequence applies:

1. A management client sets a resource on a ruleset MBean.
2. The ruleset makes a query to the MBean server to retrieve all the XU instances in the cluster. This operation requires a specific implementation for each application server.
3. A notification is sent to each instance.

Rule Execution Server deployment on WebSphere Application Server cells:

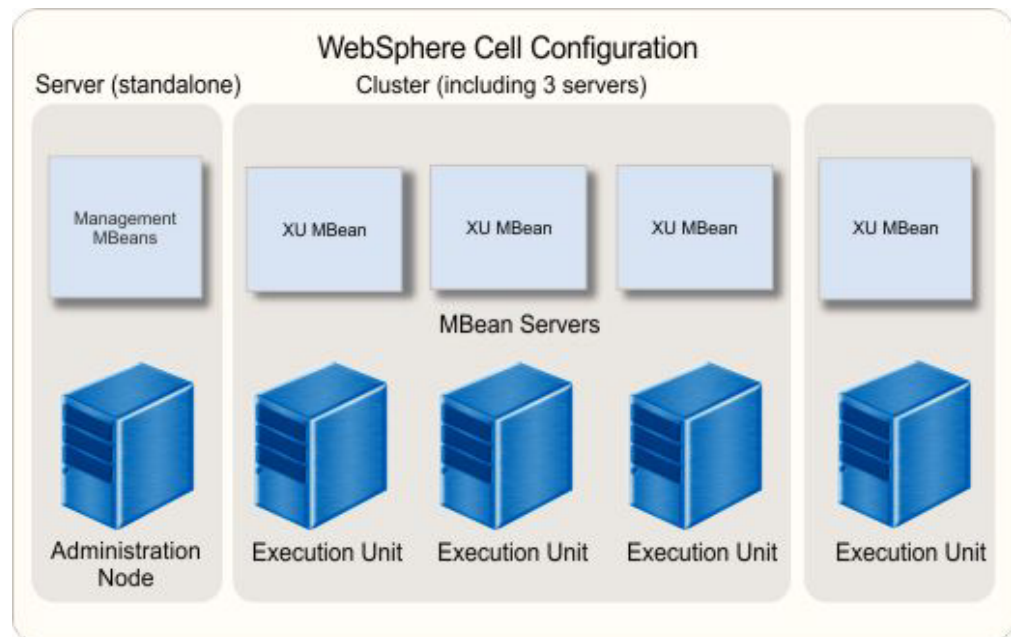
Deployment of Rule Execution Server in a WebSphere Application Server cell (distributed or z/OS) relies on a number of best practices.

A WebSphere Application Server cell is a virtual unit that consists of a deployment manager and one or more nodes. When deploying the different Rule Execution Server components on WebSphere Application Server or WebSphere Application Server for z/OS, it is important that you deploy the management module onto a single server in the cell. Only a single management module is necessary to manage all execution units (XU) on each server.

It is important to define a logical subset of servers. You must set up the configuration in the following way:

- Dedicate a stand-alone server of the cell outside the cluster to the deployment of the Rule Execution Server management module. A failover mechanism is not necessary.
- Dedicate a cluster of servers or stand-alone servers to the deployment of the XU. You can install the XU (.rar file) only at the node level and you must declare the resource adapter at the node or server level so that every server in the node contains this definition.
- Define the data source at the node level (see the documentation about creating JDBC providers).

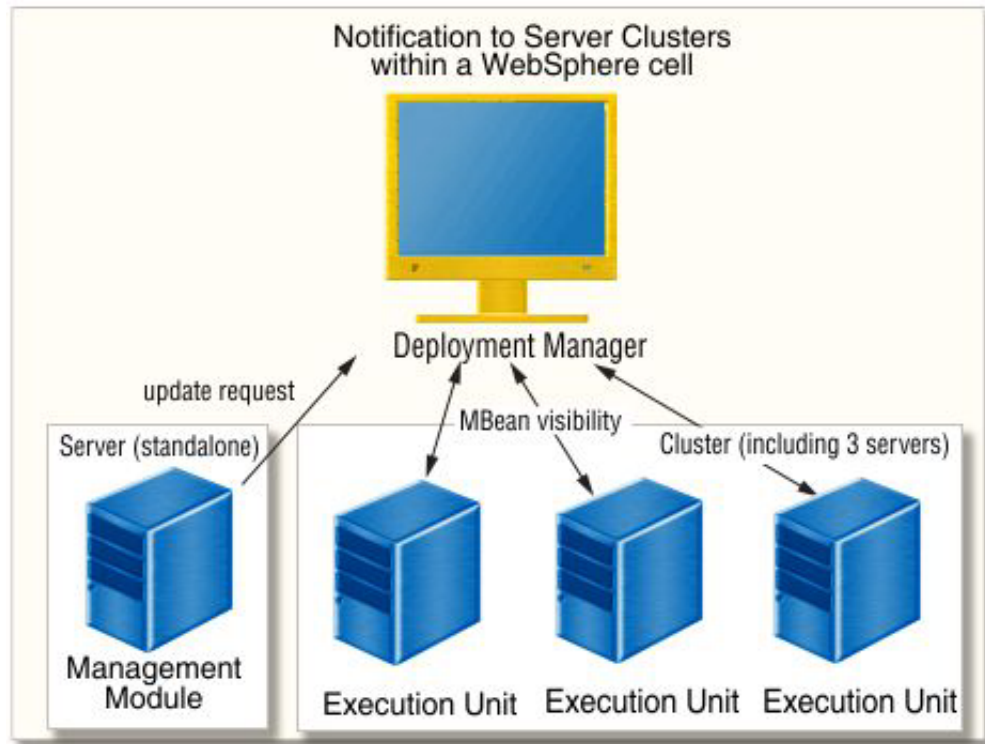
Such a configuration guarantees that a single management module is instantiated in the WebSphere Application Server cell, as illustrated in the following diagram.



The notification mechanism uses a simple pattern because Rule Execution Server MBeans rely on the distributed MBean mechanism provided by WebSphere Application Server. When an MBean is registered in a server of a WebSphere Application Server cell, it is automatically visible to the Deployment Manager MBean Server.

To notify each execution unit of changes in the management model, query the Deployment Manager MBean Server, then notify each Execution Unit retrieved by

the query.



For more information, see *Creating clusters and Build strong, flexible J2EE apps with a WebSphere cluster environment*.

Setting up WebSphere Application Server in cluster mode:

When you use WebSphere Application Server for z/OS in cluster mode, you must create a JDBC provider on each XU server and a data source for each execution unit (XU).

Procedure

1. Create a JDBC provider on each XU server in the cluster, and also one for the server that hosts the Rule Execution Server management console.
2. Create a data source for each execution unit (XU).
Each XU server must have its own data source and all data sources must point to the same database.
3. Create a data source for the Rule Execution Server management console.
The data source name can be different from resdatasource. Give data sources meaningful names such as resdatasource for serverXB1 on MVSGB. However, the data source JNDI name must be jdbc/resdatasource.

Configuring Decision Server Events on WebSphere Application Server for z/OS

To configure Decision Server Events and the event runtime for z/OS, you install the event runtime on the z/OS system and Event Designer on any platform except z/OS.

About this task

Be aware of the following limitation if you install the event runtime on a z/OS system. You can install Operational Decision Manager into a stand-alone or network deployment configuration of WebSphere Application Server. However, you cannot cluster the event runtime in Operational Decision Manager across the servers in a network deployment configuration.

Stand-alone configuration on WebSphere Application Server for z/OS:

A stand-alone configuration, also known as a single server configuration, is the simplest configuration you can use to deploy and run the event runtime in Operational Decision Manager.

A stand-alone configuration consists of a single node running an application server and one daemon server in a single z/OS system or LPAR.

The application server runs the `wberuntimeear` application (the event runtime) and the WebSphere Application Server administrative console, which you can use to configure Operational Decision Manager.

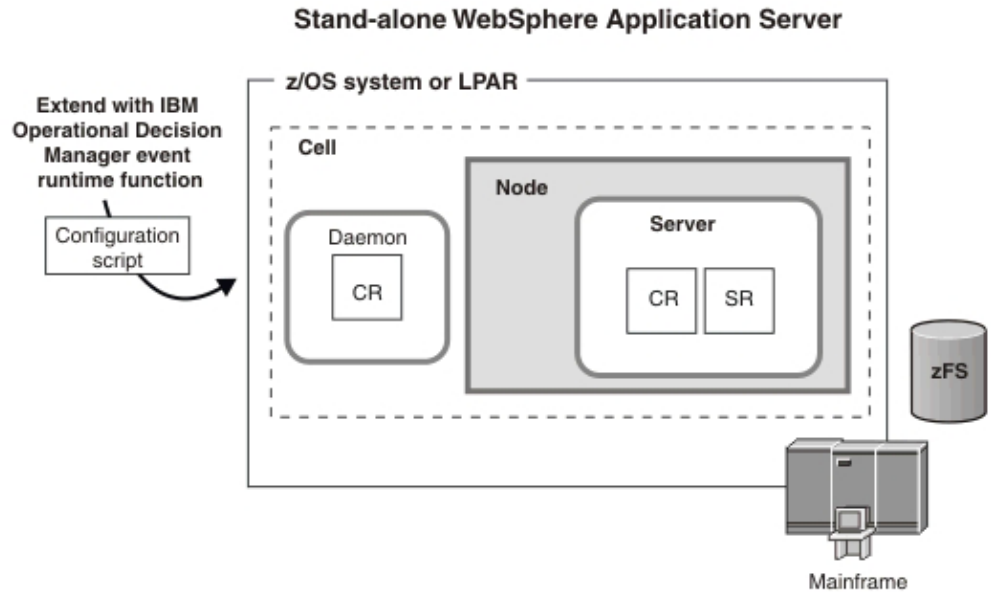
The daemon server is a unique server that runs constantly and has one controller region, which distributes server workload.

You can configure multiple application servers to run in a node. However, each server is isolated from the others. Each application server contains these items:

- Its own root file system, cell, node, TCP ports, and administration interface
- Its own administration policy for its own cell domain
- A separate flat file system to store its configuration information, in a set of XML and XMI files

Application servers in a node do not use workload distribution or common administration. You can define additional application servers in a stand-alone cell, but you cannot control them by using the WebSphere Application Server administrative console.

The following figure shows a stand-alone configuration which consists of a single node running an application server and one daemon server in a single z/OS system or LPAR:



You must have the following configuration in place before you can create a stand-alone configuration:

- Operational Decision Manager must be installed as the stand-alone server.
- Your UNIX user ID must have permission to access the UNIX shell to run the installation and augmentation scripts from inside the shell. Gaining permission to access the shell involves modifying your RACF profile and creating a home directory in the UNIX shell. The home directory is where you begin a UNIX session, and where you store environment variable files that are used when programs are run. You can also use the home directory as the main directory for storing data.
- The Operational Decision Manager product code must be loaded on to the system from tape, so that you can use it to install and configure.
- Set up the technology connectors. When running in a single server environment, the File System, HTTP, JDBC, JMS, REST, and SOAP connector applications are installed to the server where the event runtime is installed. See *Running technology connectors*.

Before you can configure and run the event runtime on Operational Decision Manager for z/OS, you must install and configure WebSphere Application Server for z/OS or Application Server Network Deployment for z/OS. If installing to Application Server Network Deployment on z/OS, the Decision Server Events application cannot be clustered.

A WebSphere Application Server stand-alone server configuration provides you with the default profile that you augment with Operational Decision Manager configuration data.

For more information about configuring WebSphere Application Server, see:

- *Creating stand-alone application server cells on z/OS using the Profile Management Tool.*
- *Building practice WebSphere Application Server for z/OS cells.*

Configuring the event runtime to run on WebSphere Application Server:

After you have installed WebSphere Application Server and loaded the Operational Decision Manager code on to the z/OS system, you must run several configuration scripts to complete the installation. For all steps in this task, check the outputs in STDOUT and STDERR to ensure successful completion. The jobs used in this task are located in ++HBRWORKDS++.SHBRJCL.

Before you begin

To run the scripts you must have authority to access and deploy to the WebSphere Application Server. Ensure that the component directory has been created if necessary; for more information see “Step 6: Creating the working directory” on page 15.

Procedure

1. Run the HBRCRTE JCL job to populate the events component directory, and check the job log to ensure that the job completed successfully. This step is required, if the file system is read-only and you are using legacy connectors that run outside WebSphere Application Server.
2. Optional. If you enabled RACF security in your WebSphere Application Server, define EJB roles for RACF security. You can use the JCL member HBREVRFC to create the EJB roles required for the event runtime when using RACF security. If you have security enabled see “Creating EJB roles and RACF groups” on page 36.
3. Run the HBREVLN JCL job to link the Operational Decision Manager files to a WebSphere Application Server installation, and to link the WebSphere Application Server default profile template directory to the Operational Decision Manager profile template.

The HBREVLN JCL job requires read and write permission to be able to link files to the WebSphere Application Server installation. The HBREVLN JCL job creates symbolic links to the *was_home/lib* directory.

Note: If the WebSphere Application Server directory is a symbolic link, you must run the WebSphere Application Server **zExpandSymlinks.sh** command from the *was_home/bin/* directory to convert the WebSphere Application Server symbolic links to physical directories. For example:

```
zExpandSymlinks.sh -configRoot <was_instance_path> -symlinks lib
```

In the example, *<was_instance_path>* is the path of the root directory for a WebSphere Application Server instance. For more information on the **zExpandSymlinks** command, see *IBM WebSphere Application Server V8 z/OS Customization*.

If the HBREVLN JCL job has already been run before, for other WebSphere Application Server instances, some links show as already existing in the job output. You must run the HBREVLN JCL job for each WebSphere Application Server instance to be configured.

4. To configure your WebSphere Application Server instance to run the event runtime, you must augment the WebSphere Application Server instance with the event runtime:
 - a. Stop WebSphere Application Server.
 - b. Run the augmentation job HBREVAUG. HBREVAUG uses the values in the HBREV member of the dataset SHBRWASC. Review the values that are contained in

HBREV before you submit the job. The values in HBREV set default values for the JMS provider. You can change these values after you configure the WebSphere Application Server instance.

Check the output of the HBREVAUG job. You can find extra logs in the `AppServer/logs/manageprofiles/` directory in the application server installation location. For example: `was_install_dir/AppServer/logs/manageprofiles/response.log` and `was_install_dir/AppServer/logs/manageprofiles/default/*.log`.

5. After a successful augmentation in step 4 on page 73, restart WebSphere Application Server. See Managing application servers.
6. After a successful augmentation after the WebSphere Application Server has restarted, use the Application Server console to set the `DISABLE_DEFERRED_CTX_REGISTRATION` property.
 - a. Go to **Servers > Server Types > WebSphere application servers > <server_name> > Transaction service > Custom properties**.
 - b. Select **New**.
 - c. Enter the name as `DISABLE_DEFERRED_CTX_REGISTRATION`.
 - d. Set the value to true.

Results

The event runtime on Operational Decision Manager for z/OS is now installed and configured.

Using Event Designer with z/OS:

To develop applications in Decision Server Events that you deploy to the event runtime on z/OS, you must install Event Designer.

About this task

Event Designer can be installed on any operating system except z/OS, so you must install these components on a second computer with a supported operating system. Event Designer can connect to the remote event runtime that is hosted on z/OS.

To install the graphical tooling, select the graphical tooling components when you install Decision Server Events on the second computer.

What to do next

When you have installed Event Designer, you can develop the assets that you require in your event application. Then you can deploy the assets to the event runtime which is hosted on a z/OS system, by specifying the TCP/IP connection details of the z/OS system when you connect to the event runtime in Event Designer.

Related tasks:

Configuring Decision Server Events on WebSphere Application Server

Verifying your Events z/OS configuration:

You can check that the event runtime on Operational Decision Manager for z/OS is properly installed and configured by loading the supplied verification event project and sending an event through the event runtime.

Before you begin

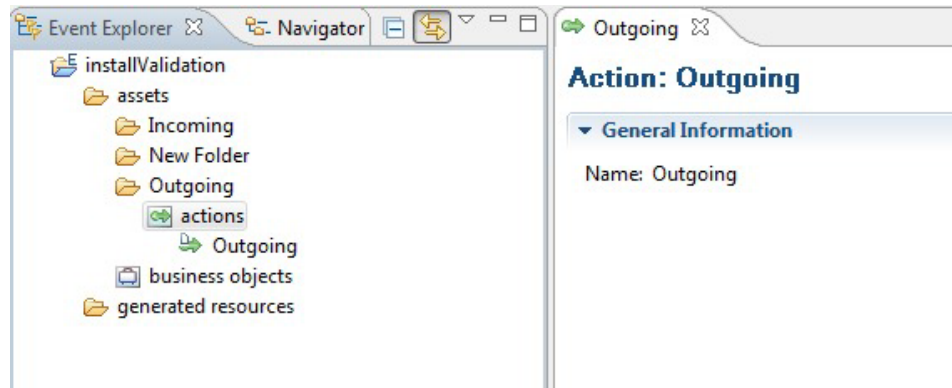
Install and configure Operational Decision Manager, including configuring a JMS messaging provider.

Ensure that you have installed Event Designer on a separate computer.

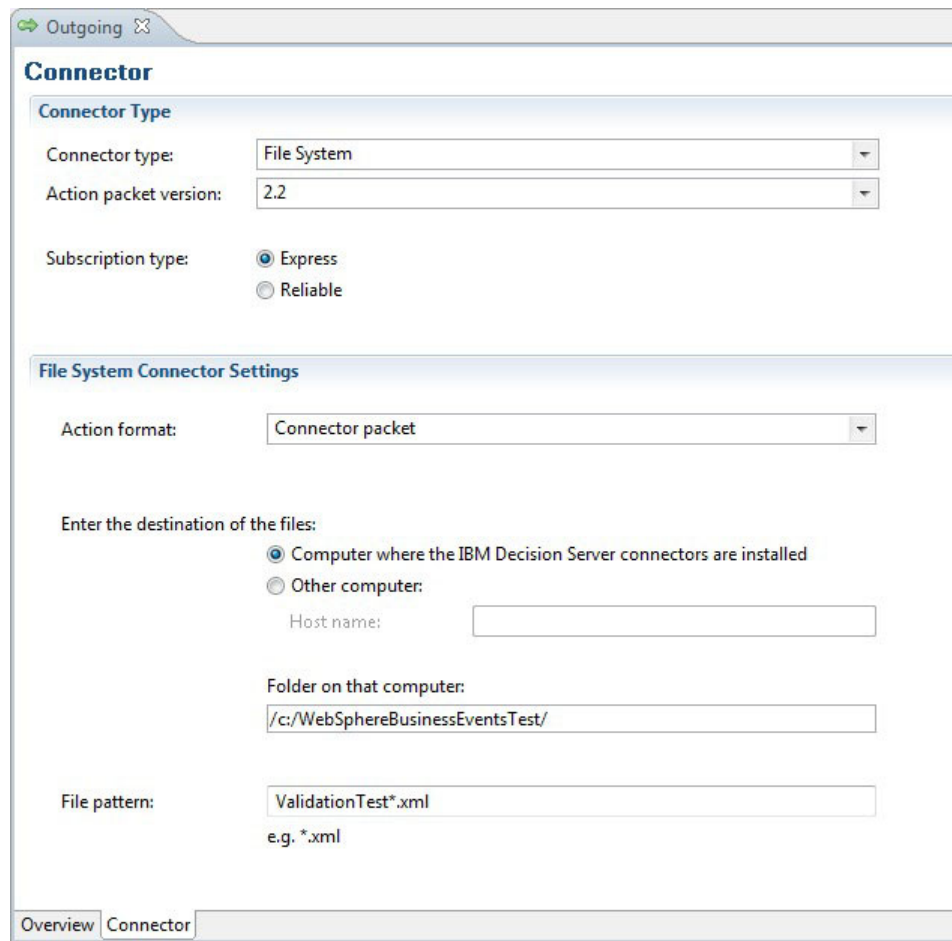
If you turned on WebSphere Application Server security during installation, you must configure user authentication to the event runtime before verifying the installation. See User authentication in the distributed platform part of this information center.

Procedure

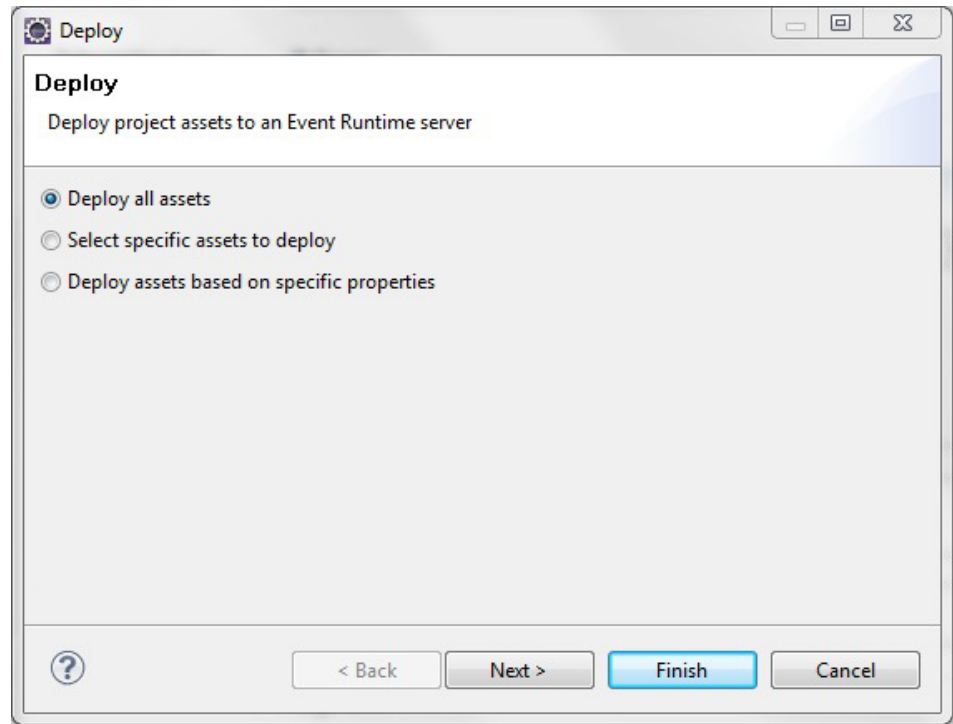
1. On the computer on which you have installed the event runtime, ensure that the WebSphere Application Server server is running.
2. Check that the event runtime application has been installed and started on WebSphere Application Server:
 - a. Start the WebSphere Application Server administrative console.
 - b. Log in to the WebSphere Application Server administrative console with a user ID of your choice.
 - c. In the navigation tree of the WebSphere Application Server administrative console, click **Applications > Application Types > WebSphere enterprise applications**.
 - d. Ensure that the wberuntimeear application is listed and shown as started.
 - e. If you have prepared the environment for testing, ensure that the wbetesterear application is listed and shown as started.
3. On the computer on which you installed the event runtime, create a directory called `DecisionServerTest` and make a note of the location in which you created the directory. Note that the directory location created must allow the ID that the WebSphere Application Server server runs under to write to this location.
4. On the computer on which you installed Event Designer, open Event Designer and right-click in the Event Explorer view, then click **Import**. The Import wizard opens.
5. In the Import wizard, click **Event Project from XML File > Next**.
6. Click **Browse**, navigate to `<InstallDir>\connectors\validation` and select the event project called `installValidation.xml`. Click **Next**. Accept the default settings for the rest of the wizard and click **Finish** to import the event project into your workspace.
7. In the Event Explorer view, expand the event project, expand the **Outgoing** folder, and open the Outgoing action:



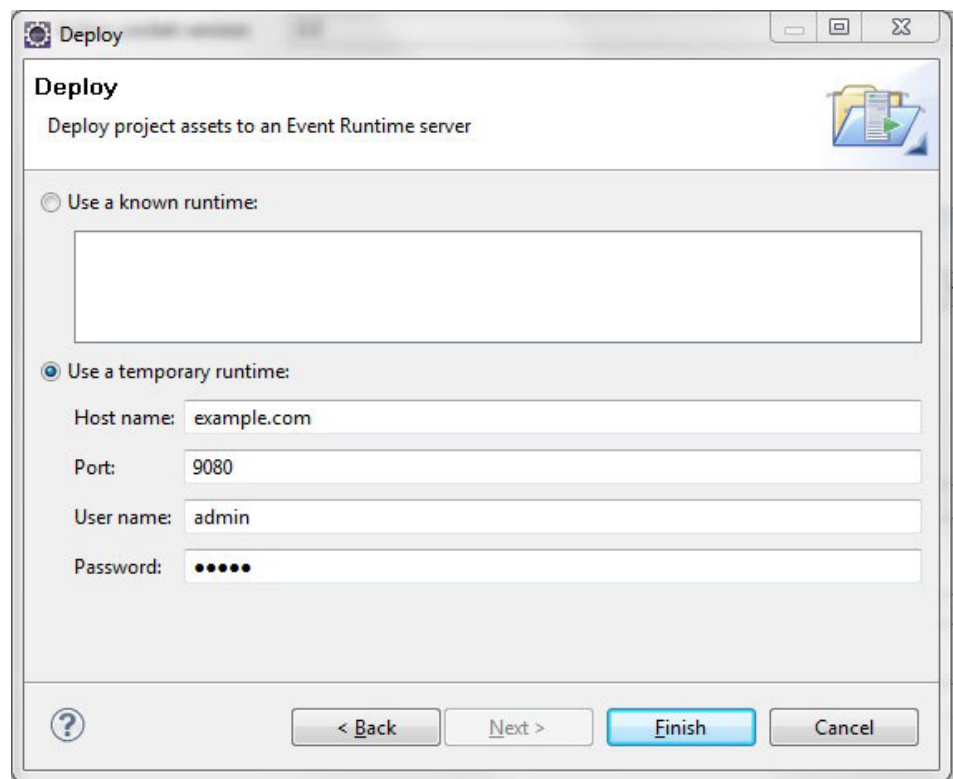
8. In the Action editor, click the **Connector** tab:



9. In the **Folder on that computer** field, replace the existing value with the complete path that points to the location of the `DecisionServerTest` directory that you created. Save your changes.
10. To deploy the event project, right-click the event project in the Event Explorer view and select **Deploy**. The Deploy wizard opens:



11. Select **Deploy all assets** and click **Next**.
12. Enter the connection details for your event runtime:



Ensure that the host name and port values are correct for the WebSphere Application Server instance in which the event runtime is deployed. The port

value is the WC_defaulthost port, which can be determined through the WebSphere Application Server administrative console. Enter the user ID and password, if security is enabled.

13. Click **Finish** to deploy the event project. A message is displayed indicating that the event project is deployed successfully. Close Event Designer.
14. Send a test event to the event runtime by running the Command-line connector:
 - a. If you have RACF security turned on in WebSphere Application Server, you must do some additional tasks so that the technology connectors start. For more information see Starting technology connectors when WebSphere Application Security is enabled. If RACF security is enabled in WebSphere Application Server, you must run the HBRVIVP JCL job with a user ID and a password for example: `cmdline.sh -uid <user> -pwd <password> +`, where `<user>` is replaced by the user ID of the person authorized to make this call, and `<password>` is the password of the user .

In addition, the user ID might require connecting to the key ring file, for example:

```
RACDCERT ADDRING(<WASKeyring>) ID(<user>) RACDCERT ID(<user>)
CONNECT (RING(<WASKeyring>) LABEL('WebSphereCA') CERTAUTH
```

If necessary, ask your security administrator to configure this security requirement.

- b. Submit the JCL job HBREIVVP, which uses a sample event contained in the `installValidation_event.xml` file. Here is an example of the job:

```
//HBREVLN EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//STDENV DD *
WASADMINHOST=<host_name>
WASBOOTSTRAPPORT=<port_number>
WBE_WAS_HOME=/WebSphere/<was_name>/AppServer
WBE_ZOS=yes
/*
//SYSTSIN DD *
BPXBATCH SH +
export WBE_INSTALL=+
/usr/lpp/zDM/V8R6M0/+
events; +
/usr/lpp/zDM/V8R6M0/+
events/connectors/bin/cmdline.sh -uid myid -pwd mypsw +
/usr/lpp/zDM/V8R6M0/events/+
connectors/validation/installValidation_event.xml
/*
```

A new file called `ValidationTest random-number.xml` is put in the `DecisionServerTest` directory, where `random-number` is a system-generated number that makes the file name unique.

Submitting the HBREIVVP job might result in 0, but you must always check the output of the HBREIVVP job in the job log.

Results

You have now verified your z/OS installation.

Tuning your z/OS configuration:

Some tuning of the system might lead to higher performance when processing events.

About this task

When considering how you tune your system, there are two main areas to look at:

- The event runtime and WebSphere Application Server
- JMS messaging.

The event runtime and WebSphere Application Server

- Turn off logging. For more information, see Logging information in the distributed platform part of this information center.
- Turn off the recording of history, if not required. See Configuring the event runtime to record history in the distributed platform part of this information center.
- Use technology connectors only if required.
- Ensure that there are sufficient event rule processing threads. If you are unable to achieve full processor utilization on the Decision Server Events server, consider increasing the value of this property:
`as.director.server.ruleProcessorInstances`.
- On z/OS, using the Type 2 JDBC driver to connect to a local DB2 database performs better than the Type 4 driver. To modify the driver type use the WebSphere Application Server administrative console and navigate to **Resources > JDBC > Data sources > WBE Datasource**. You can modify the Driver type on the **WBE Datasource** page.

Note: To use the Type 2 driver you might have to set the **Native library path** on the JDBC Provider and add the DB2 SDSNEXIT, SDSNLOAD, and SDSNLOAD2 libraries to the Servant procedure STEPLIB.

- Tune the database.
 - See DB2 tuning tips for z/OS.
- Tune the JVM.
 - Consider tuning the JVM in the Servant region and the JVM in the Adjunct region.
 - Key tuning parameters are JVM heap size and garbage collection policy. A small heap can cause frequent garbage collection, which can degrade performance. A large heap can result in longer pause times.
 - See Tuning the JVM.

JMS messaging

- For persistent messaging, consider using a zFS file system, as this offers improved performance to HFS.
- The key tuning parameters relate to the choice of message reliability level, activation specifications, and size of the discardable data buffer. See Tuning messaging performance with service integration.
- Message reliability level: A lower reliability level generally provides a higher throughput. The default reliability level for non-persistent JMS messages is Express Non-Persistent, and for persistent JMS messages it is Reliable Persistent. The reliability level can be changed by using the connection factories. Use the WebSphere Application Server administrative console (for example, **Resources > JMS > TopicConnectionFactory > WbeTopicConnectionFactory**).
- Activation specification: Particularly for non-durable JMS events, consider delivering the messages in batches from the input topic to Decision Server Events. This can deliver events more efficiently to Decision Server Events. For

example, using the WebSphere Application Server administrative console, navigate to **Resources > JMS > Activation specifications > wbe_events**, and set Maximum batch size.

- Bypass the connectors and send events directly to Operational Decision Manager by configuring WebSphere MQ to be the JMS provider. Performance improves when events are received from the WebSphere MQ queue. See *Configuring WebSphere MQ to be the JMS provider* in the distributed platform part of this information center.

Configuring additional options for Decision Server Events for z/OS:

You can customize the behavior of the event runtime by modifying the Decision Server Events properties. For example, you can change the messaging provider that is used by the event runtime.

Procedure

1. Customize the event runtime properties. See *Setting Properties* in the distributed platforms section of the Operational Decision Manager information center.
2. Configure the event runtime to use WebSphere MQ as the JMS messaging provider. The event runtime can be configured to work with only one JMS provider: either the WebSphere Application Server default messaging, or WebSphere MQ, but not both. See *Configuring WebSphere MQ to be the JMS provider* in the distributed platforms section of the Operational Decision Manager information center.
3. Edit the setenv file to add the appropriate values for the WebSphere Application Server instance on which Operational Decision Manager is going to run. The .sh file that configures the messaging provider relies on settings in the setenv file. Check and confirm your environment variables settings before running the script. See *Editing the setenv files* in the distributed platforms section of the information center.

Configuring additional options for the File System Connector:

If you use the File System Connector on z/OS, you might notice that the wfileconnector application does not start when you start the WebSphere Application Server. You can resolve the problem by setting a parameter.

About this task

To resolve the problem, set the `com.ibm.ejs.j2c.J2CServerListener.deferEndpointActivation` property to true to defer endpoint initialization until after the Application Server servant region has started. It should be added as a JVM Custom Property for the Application Server servant and control region through the administrative console.

Procedure

1. Go to **Servers > Application Servers > server_name > Process Definition > Servant | Control > Java Virtual Machine > Custom Properties**.
2. Click **New** and use the following values for the new property:
 - Name: `com.ibm.ejs.j2c.J2CServerListener.deferEndpointActivation`
 - Value: true
3. Click **OK**, then save and synchronize the changes. Restart the server for the changes to take effect.

Uninstalling on z/OS:

Remove the event runtime configuration and product code.

Before you begin

Ensure that you have completed the steps in Editing the setenv files, which is found in the distributed platform part of this information center.

About this task

Run the JCL job HBREVUNI and use the HBREVUN member to pass information to the job.

Procedure

1. Review the HBREVUN member in the SHBRWASC dataset to make sure the environment and site variables are set correctly. The members of SHBRWASC contain properties files for configuring Decision Server Events.
2. Run the JCL job HBREVUNI.

Results

The event runtime configuration and product code is now removed from the system.

Configuring the Decision Center consoles on WebSphere Application Server for z/OS

After you set up security on the application server, you must then choose how you configure the Decision Center consoles on an instance of WebSphere Application Server for z/OS . You can do this task by using wsadmin scripts or by following the manual steps.

Automated configuration of Decision Center on WebSphere Application Server for z/OS:

One way to configure the Decision Center consoles on WebSphere Application Server for z/OS is to run the provided wsadmin scripts, which can be used to complete the deployment and configuration steps.

Before you start:

You can configure Decision Center on WebSphere Application Server for z/OS with DB2 by running a set of scripts.

Important:

During the execution of the **wsadmin** scripts, WebSphere Application Server might be restarted. If this is a problem, perform a manual configuration.

The process is as follows:

1. Complete the prerequisites (authorize the load library, customize the variables, decide on a topology, create the working data sets and directory, configure DB2, and configure security):
 - a. Configuring.
 - b. Configuring security on WebSphere Application Server for z/OS

2. Run the JCL job as indicated in “Submitting the JCL jobs.”
3. Deploy the MBean descriptors.

Submitting the JCL jobs:

Submit the JCL jobs to complete the Decision Center configuration.

About this task

HBRDCWAS is the job that configures Decision Center on WebSphere Application Server for z/OS. After you configured the JCL scripts in the SHBRJCL data set as explained in Configuring, you can submit that job to configure Decision Center for z/OS.

Procedure

1. Submit the JCL jobs in the following order:
 - a. HBRDCWAS
 - b. HBRSDVVS

When each job is finished, a build successful message is displayed.

2. Check the job spool **STDOUT** and **STDERR** for the logs.

If a job fails, you can find the WebSphere Application Server installation properties in the data set members ++HBRWORKDS++.SHBRWASC(HBRRTS) and ++HBRWORKDS++.SHBRWASC(HBRDVS), which are created during “Step 6: Creating the working directory” on page 15.

Results

Upon successful completion of the job, the Decision Center consoles are configured to WebSphere Application Server for z/OS.

What to do next

- If you want to deploy rules to Rule Execution Server or test rules by using Decision Validation Services, you must also deploy the MBean JAR file, see “Deploying the Rule Execution Server MBean descriptors” on page 42.
- To verify and complete the configuration, go to Verifying your configuration and follow the steps up until running the diagnostics.
- To learn about testing and simulating rulesets in Rule Designer and Decision Center, see Testing with Decision Validation Services in the distributed platform part of this information center.

Setting custom properties for Decision Center security:

To ensure that the Decision Center consoles work correctly, set the `InvalidateOnUnauthorizedSessionRequestException` and `setContextRootForFormLogin` properties in WebSphere Application Server.

Before you begin

To set custom properties for security, you work in the WebSphere Integrated Solutions Console. Make sure that you have a WebSphere Application Server profile start the server, and then open the console and log in by using the user ID and password that you defined in the profile.

About this task

To prevent potential session and authentication errors, set the following custom properties in the WebSphere Integrated Solutions Console.

Location in the administrative console	Property and value	Description
Application servers > server1 > Session management > Custom properties	invalidateOnUnauthorizedSessionRequestException = true	If the same user ID accesses the Business console and the Enterprise console at the same time, this property prevents authentication errors. For more information, see the following technote: http://www.ibm.com/support/docview.wss?uid=swg21609826 .
Global Security > Custom properties	com.ibm.websphere.security.setContextRootForFormLogin = true	If the Business console and the Enterprise console are used on the same application server, this property prevents a WebSphere Application Server cookie from pointing to the incorrect application. For more information, see the following technote: http://www.ibm.com/support/docview.wss?uid=swg1PM58885 .

Manual configuration of Decision Center on WebSphere Application Server for z/OS:

One way of configuring the Decision Center consoles on an instance of WebSphere Application Server for z/OS is to manually do the steps one by one.

Before you start:

Before you proceed with the configuration, review the tasks that you must perform to configure Decision Center on WebSphere Application Server for z/OS.

Before you start a WebSphere Application Server configuration, you must complete the steps in “Configuring,” on page 1.

As an alternative to using the WebSphere Application Server Integrated Solutions Console, you can also use the customizable **wsadmin** scripts. For more information, refer to Automated configuration of Rule Execution Server on WebSphere Application Server for z/OS.

A specific integration extension for the IBM Process Server platform is available. For more information, see WebSphere Operation Decision Management Integration SupportPacs.

After your installation is finished, Decision Center is ready to use (see Opening the Decision Center consoles in the distributed platform part of this information center). The first time that you open Decision Center, it does not contain a rule project. You must publish a project from Rule Designer (see Publishing a project to Decision Center).

It is assumed that you are using a WebSphere Application Server for z/OS profile that has administrative security enabled.

Software Prerequisites

You must install the following software before you configure Decision Center:

- WebSphere Application Server V8 for z/OS
- Decision Center for z/OS Version 8.6









Note: WebSphere Application Server OEM Edition prevents the installation of user-written web applications. This restriction prevents you from customizing and repackaging the Scenario Service Provider (SSP), and packaging the run time for testing purposes in Decision Center. You cannot use Decision Validation Services with WebSphere Application Server OEM Edition.


You also need the following rights:

- Access to the WebSphere Integrated Solutions console
- Authority to start and stop WebSphere Application Server for z/OS

Configuration steps

The following table summarizes the steps that you follow to configure Decision Center on WebSphere Application Server for z/OS.

Step	Required
"Step 1: Configuring security on WebSphere Application Server" on page 85	
"Step 2: Creating a JDBC provider" on page 86	
"Step 3: Creating a data source and connection pool" on page 87	
"Step 4: Creating J2C authentication data" on page 88	
"Step 5: Setting the component and container managed aliases for the data source" on page 88	
"Step 6: Setting the currentSchema parameter" on page 88	
"Step 7: Testing the connection to the database" on page 89	
"Step 8: Deploying the EAR file on WebSphere Application Server" on page 89	

Step	Required
"Step 9: Changing the class loading sequence" on page 90	
Verifying your configuration of Decision Center	Recommended

Step 1: Configuring security on WebSphere Application Server:

When you choose to configure Decision Center on WebSphere Application Server for z/OS by using DB2, the first step consists in setting up administrative security and application security.

Decision Center access is managed by the application server security. To access Decision Center in WebSphere Application Server for z/OS, you must define a user registry as explained in "Configuring security on WebSphere Application Server for z/OS" on page 34. Two examples are provided, one using RACF, the other using a federated repository.

Updating the security policies:

You must override security policies for the Decision Center console so that it can record and manage a set of MBeans.

After the global security of WebSphere Application Server for z/OS is activated, the MBean server cannot be accessed from the deployed application. You must override these security policies for the Decision Center console because the console needs to record and manage a set of MBeans.

Decision Center is packaged with a specific policy file that overrides the server policies. The `was.policy` file is packaged in the `jrules-teamserver-WAS<version_number>.ear` file, located in the META-INF directory.

When you enable Java 2 Security on WebSphere Application Server for z/OS, you must update the `was.policy` file to give read and write permissions on each directory where your RuleDocs are published, similar to the following example:

```
permission java.io.FilePermission "<path to my ruledoc folder>${/}-",
    "read, write, delete";
```

If you do not update the `was.policy` file, users will not be able to synchronize RuleDocs on the file system.

Setting custom properties for Decision Center security:

To ensure that the Decision Center consoles work correctly, set the `invalidateOnUnauthorizedSessionRequestException` and `setContextRootForFormLogin` properties in WebSphere Application Server.

Before you begin

To set custom properties for security, you work in the WebSphere Integrated Solutions Console. Make sure that you have a WebSphere Application Server profile start the server, and then open the console and log in by using the user ID and password that you defined in the profile.

About this task

To prevent potential session and authentication errors, set the following custom properties in the WebSphere Integrated Solutions Console.

Location in the administrative console	Property and value	Description
Application servers > server1 > Session management > Custom properties	invalidateOnUnauthorizedSessionRequestException = true	If the same user ID accesses the Business console and the Enterprise console at the same time, this property prevents authentication errors. For more information, see the following technote: http://www.ibm.com/support/docview.wss?uid=swg21609826 .
Global Security > Custom properties	com.ibm.websphere.security.setContextRootForFormLogin = true	If the Business console and the Enterprise console are used on the same application server, this property prevents a WebSphere Application Server cookie from pointing to the incorrect application. For more information, see the following technote: http://www.ibm.com/support/docview.wss?uid=swg1PM58885 .

Step 2: Creating a JDBC provider:

Create a JDBC provider in WebSphere Application Server for z/OS to enable Decision Center.

About this task

The first step in creating a data source is to create a JDBC provider.

Procedure

1. Log in to the Integrated Solutions Console.
2. Define the JDBC driver path.
 - a. Navigate to **Environment > WebSphere variables**.
 - b. Click **DB2UNIVERSAL_JDBC_DRIVER_PATH**.
 - c. Type the directory location for db2jcc.jar, and db2jcc_license_cisuz.jar in the **Value** field. For example: /usr/lpp/db2910/classes
 - d. Click **Apply**, then click **Save**.
3. Define the JDBC driver native path.
 - a. Click **DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH**.
 - b. Type the Native library path in the **Value** field. For example: /usr/lpp/db2910/lib
 - c. Click **Apply**, then click **Save**.
4. In the side panel, open **Resources > JDBC** and click **JDBC Providers**.
5. In the **Scope** side panel, select **Node=xxx**, **Server=yyy**, where *xxx* is the name of your node and *yyy* the name of your server, and click **New**.
6. In **Step 1**, set the values for the following parameters, then click **Next**.
 - Database type: DB2

- Provider type: DB2 Universal JDBC Driver Provider
 - Implementation type: Select **Connection pool data source**, then type a name, for example, DB2 Universal JDBC Driver Provider for Decision Center
7. In **Step 2**, click **Next**. A summary is provided in **Step 3**.
 8. Click **Finish** and **Save** to save the changes directly to the master configuration.

Step 3: Creating a data source and connection pool:

Create a connection pool and a data source in WebSphere Application Server to enable Decision Center.

About this task

After you have created your JDBC provider, you can create a data source and connection pool.

If WebSphere Application Server is used in cluster mode, you must define the data source at node level in the cluster (as opposed to cluster level). See Rule Execution Server deployment on cluster environments.

Procedure

1. In the Integrated Solutions Console, open **Resources > JDBC** and click **Data sources**.
2. In the **Scope** side panel, select the scope that you selected for the JDBC provider in “Step 2: Creating a JDBC provider” on page 86 and click **New**.
3. In **Step 1**, enter a name for the data source in the **Data source name** field and `jdbc/<your_data_source_name>` in the **JNDI name** field, and click **Next**.
4. In **Step 2**, select the JDBC provider that you created in “Step 2: Creating a JDBC provider” on page 86, and click **Next**.
5. In **Step 3**, enter the specific database properties for the data source:

Driver type

4

Database name

Type your database name, for example RTSDB.

Server name

Type your server name.

Port number

Type your port number.

6. Clear the **Use this data source in container managed persistence (CMP)** option and click **Next**.
7. In **Step 4**, click **Next**.
Step 5 displays a summary of your settings.
8. Click **Finish**.
The connection pool is created and associated with the data source.
9. Click **Save** to save the changes directly to the master configuration.

Step 4: Creating J2C authentication data:

After you have created your data source and connection pool, you create the J2C authentication data. J2C is a secure mechanism for integrating enterprise information systems to an application server and enterprise applications.

About this task

This step uses the user ID and password (user=rtsAdmin, password=rtsAdmin). Your user ID and password might be different, depending on the user or schema that is used.

Procedure

To create J2C authentication data:

1. In the Integrated Solutions Console, open **Resources** > **JDBC** and **Data sources**.
2. Click the name of the data source, for example **Decision Center Data Source**.
3. Under **Related Items**, click **JAAS - J2C authentication data**.
4. Click **New** and set the fields **Alias**, **User ID**, and **Password**. For example, you can enter the following values:
 - **Alias:** rtsDB2user
 - **User ID:** rtsAdmin
 - **Password:** rtsAdmin
5. Click **Apply** and **Save** to save directly to the master configuration.

Step 5: Setting the component and container managed aliases for the data source:

After you have defined the rtsDB2user alias for J2C authentication data, you must set the aliases on the Decision Center data source.

Procedure

1. Open **Resources** > **JDBC** > **Data sources** and click the name of the data source, for example **Decision Center Data Source**.
2. In the **Security settings** section, for **Component-managed authentication alias**, select the <node name>/rtsDB2user alias.
<node name> is the name of the WebSphere Application Server node on which you are installing Decision Center and rtsDB2user is the alias that you defined in "Step 4: Creating J2C authentication data."
3. Select <node name>/rtsDB2user for the **Container-managed authentication alias**.
4. Click **Apply** and **Save** to save directly to the master configuration.

Step 6: Setting the currentSchema parameter:

Set the **currentSchema** parameter for the Decision Center data source.

Procedure

1. Open **Resources** > **JDBC** > **Data sources** and click **Decision Center Data Source**.
2. Under **Additional Properties**, click **Custom properties**.

3. Click **New** to add a property and set the **Name** to be **currentSchema** and the **Value** to be the schema that is used to access your Decision Center tables. For example, HBR1.
4. Click **Apply** and **Save** to save directly to the master configuration.

Step 7: Testing the connection to the database:

Test the connection to your database.

About this task

After you have created a data source and connection pool, test the connection to your database.

Procedure

1. In **Data Sources**, select the check box next to **Decision Center Data Source**.
2. Click **Test Connection**.

The status of the connection is indicated at the top.

Step 8: Deploying the EAR file on WebSphere Application Server:

At this point, you can deploy the Decision Center EAR file on WebSphere Application Server.

About this task

Important:

- When you deploy the Decision Center EAR file, the process sets the persistence locale. After you have saved a rule to the database, you are no longer allowed to change the persistence locale. If you want to install Decision Center in a language other than English, take note of the instructions provided in Set the persistence locale in the distributed platform part of this information center.
- If you redeploy the Decision Center EAR file, your action has the following consequences:
 - The class loading sequence is lost. See “Step 9: Changing the class loading sequence” on page 90. WebSphere Application Server reverts to the default parent first setting.
 - All users, such as rtsUser1, rtsAdmin, lose their role, even though they belong to the correct group. When you sign in to the Decision Center console, a message is displayed, such as rtsUser1 does not have the correct role.

Procedure

1. In the Integrated Solutions Console, click **Applications > New Application** and then **New Enterprise Application**.
2. Click **Browse** and navigate to the Decision Center following EAR file, then click **Next**.
`<InstallDir>/teamserver/applicationservers/WebSphere<version_number>/jrules-teamserver-WAS<version_number>.ear`
3. Select the check box next to **Detailed - Show all installation options and parameters**.
4. Expand **Choose to generate default bindings and mappings**, select the check box **Generate Default Bindings**, then click **Next**.
5. Click **Continue** to accept the security warning.

6. In **Step 1**, click **Next** to accept the default settings.
7. In **Step 2**, select the target server and click **Next**. Keep the default setting if you have only one server.
8. For **Step 3 to Step 8**, click **Next** to accept the default settings.
9. Optional: Unless you use a RACF repository, click **Step 9: Map security roles to users and groups** to map the security roles to users and groups as follows.
The application server uses the roles that are defined in the deployment descriptors. You must map these to the groups found in the security settings. Ignore this step if you use a RACF repository.
 - a. Select a check box next to a role in the table and click **Map groups**.
 - b. Click **Search** in the middle of the page to display the groups.
 - c. Map the group to the role that you are editing by moving that group to the **Selected** column, then click **OK**.
 - d. Repeat 9a to 9c for all the roles.

After you have completed the assignments, they are shown as follows:

Role	Special subjects	Mapped users	Mapped groups
rtsUser	None		rtsUser
rtsAdministrator	None		rtsAdministrator
rtsConfigManager	None		rtsConfigManager
rtsInstaller	None		rtsInstaller
Validator	None		Validator
Eligibility	None		Eligibility

10. Click **Next** to get to the Summary, then click **Finish**.
11. After the installation has completed, click **Save** to save your workspace changes to the master configuration.

Note:

If the database has not been defined already, an error message informs you that you must run the database creation jobs before proceeding. However, you can use the Installation Settings wizard to import extension files.

Step 9: Changing the class loading sequence:

Before you can deploy the EAR file, you must set the class loading sequence to parent last.

About this task

After deploying the Decision Center EAR file, you must set the class loading sequence. The Decision Center application does not support the default parent first configuration.

Procedure

1. In the side panel, open **Applications > Application Types > WebSphere enterprise applications**.
2. Click **ILOG Decision Center** or the name that you specified for the Decision Center application.
3. In the **Modules** section, click **Manage Modules**.

4. Click `teamservice` or `teamservice-WAS8` and select the **Class loader order > Classes loaded with local class loader first (parent last)**, then click **OK**.
5. Click **OK** to save changes directly to the master configuration.
6. On WebSphere Application Server 8 only, repeat steps 3 on page 90, clicking `decisioncenter` this time, to 5.
7. In the side panel, open **Applications > Application Types > WebSphere enterprise applications**.
8. Select the check box next to **ILOG Decision Center** (or the name that you specified for the Decision Center application) and click **Start** to start the application.

What to do next

You can now verify the deployment of Decision Center as explained in Verifying your configuration of Decision Center in the distributed platform part of this information center, and follow the steps up until running the diagnostics.

Additional steps to configure Decision Validation Services:

Optionally, you can deploy, configure, and test Decision Validation Services to complement your Decision Center configuration on zRule Execution Server for z/OS or on WebSphere Application Server for z/OS.

Configuring Decision Validation Services on WebSphere Application Server for z/OS:

You can deploy and configure Decision Validation Services to complement your Decision Center configuration on WebSphere Application Server for z/OS.

Note:

- Decision Validation Services uses the `resdatasource` rather than the Decision Center `ilogdatasource`. Before you configure the Scenario Service Provider (SSP) you must configure the execution unit (XU) RAR. For more information, see *Configuring Rule Execution Server on WebSphere Application Server for z/OS*.
- If you intend to use Decision Warehouse to store your test execution results, you must create a dedicated schema in your database containing these tables and views. For more information, see *Configuring a DB2 database as the persistence layer for Rule Execution Server on WebSphere Application Server for z/OS*.
- In the subsequent configuration steps, `<version_number>` refers to the version of WebSphere Application Server for z/OS to which you deploy the SSP archive.

Deploying the default SSP archive to test Decision Validation Services:

Deploy the Scenario Service Provider (SSP) archive to test the Decision Validation Services application on WebSphere Application Server for z/OS.

About this task

You can use this procedure to deploy the default SSP archive or any subsequent deployment of a repackaged archive. Deploy the default SSP EAR if you want to check the availability of the feature. You can also deploy the default SSP EAR file and use this to test your rules if you have an XML XOM. An XML XOM is included in the ruleset archive inside a RuleApp, and therefore the SSP EAR file does not have to be repackaged to include the XOM.

Note:

The Scenario Service Provider (SSP) archive must be deployed on the same server as the execution unit (XU).

If you are installing Decision Validation Services on z/OS, you can use customizable installation scripts to perform this task. For more information, see Automated configuration of Rule Execution Server on WebSphere Application Server for z/OS.

Procedure

1. Open the Integrated Solutions Console.
2. In the side panel, open **Applications** and click **WebSphere Enterprise Applications**.
3. In the side panel, click **Install**.
4. In the side panel, select **Local file system** and **Browse** to the following archive, then click **Next**.`<InstallDir>/executionserver/applicationservers/WebSphere<version_number>/jrules-ssp-WAS<version_number>.ear`
5. Select the check box **Detailed - Show all installation options and parameters**.
6. Expand **Choose to generate default bindings and mappings** and select the check box **Generate Default Bindings**.
7. Click **Next**, and then click **Continue** to accept the security warning.
8. Continue through the wizard pages as follows:
 - WebSphere Application Server 8: In **Step 1** through **Step 9**, click **Next** to accept the default settings.
9. In **Step 9** (for WebSphere Application Server 7) or **Step 10** (for WebSphere Application Server 8), do one of the following:
 - If you have activated security using a federated repository, select **resAdministrators** and click **Map groups**.
 - Otherwise, go to step 14.
10. Click **Search**.
11. Select the **resAdministrators** group. To do this, select it under **Available** and click the arrow button to move it to the **Selected** column.
12. Click **OK** to return to the Map security roles to users or groups page.
13. Repeat steps 9 to 12 for the **resDeployers** group, selecting **resDeployers**.
14. Complete the installation as follows:
 - WebSphere Application Server 8: Click **Next** until a summary is displayed on **Step 13**, then click **Finish**.
15. After the installation has completed, click **Save** to save directly to the master configuration.
16. In the side panel, open **Applications** > **Application Types** > **WebSphere enterprise applications**, and click **jrules-ssp-WAS<version_number>**.
17. Click **Manage Modules**.
18. Click **Scenario Service Provider**.
19. Under **General Properties**, for **Class loader order**, select **Classes loaded with local class loader first (parent last)**.
20. Click **OK**.
21. Click **OK** again and then **Save** to save directly to the master configuration.

22. In the side panel, open **Applications** > **Application Types** > **WebSphere enterprise applications**.
23. In the **Enterprise Applications** page, select the check box next to **jrules-ssp-WAS<version_number>** and click **Start** to start the application.
24. To check on the availability of Decision Validation Services, enter the URL `http://<host>:<port>/testing` in a browser.

When you log in as an authorized role, the SSP server displays a home page which contains some information about the SSP server. The About screen provides more information after you first run a test or simulation:

- RuleSession: whether it is a POJO or Java SE rule session
- DAO Factory Class: the DAO (Data Access Object) factory class used to store the execution trace into Decision Warehouse.
- Job store class: the name of the class used to store the Decision Validation Services job into a cache to free the memory during long computations.
- Job pool size: the size of the pool for asynchronous execution.

Deploying a repackaged SSP archive:

Redeploy a Decision Validation Services archive.

About this task

You can use this procedure to redeploy the default Scenario Service Provider (SSP) archive or any subsequent deployment of a repackaged archive. After you have tested the feature using the default SSP, you can update the artifact with your specific configuration and XOM.

You can use customizable installation scripts to perform this task. For more information, refer to *Configuring Rule Execution Server on WebSphere Application Server for z/OS using wsadmin scripts*.

The following procedure shows how to redeploy the SSP archive using the WebSphere Application Server administrative console.

Procedure

1. Open the Integrated Solutions Console.
2. In the side panel, open **Applications** and click **WebSphere Enterprise Applications**.
3. Select the deployed SSP application in the side panel and click **Uninstall**.
4. Click **OK** to confirm uninstallation of the SSP application.
5. Click **Save** to save the changes directly to the master configuration.
6. To deploy the new SSP, follow the procedure in “Deploying the default SSP archive to test Decision Validation Services” on page 91.

Configuring Decision Validation Services on zRule Execution Server for z/OS:

You can add your Scenario Service Provider (SSP) on zRule Execution Server for z/OS to enable users of Decision Center to test and simulate rules using z/OS native data (COBOL data or PL/I data).

Note:

If you intend to use Decision Warehouse to store your test execution results, you must create a dedicated schema in your database containing these tables and views. For more information, see *Configuring a DB2 database as the persistence layer for zRule Execution Server for z/OS*

Adding the SSP to zRule Execution Server for z/OS:

To run tests and simulations using a SSP (Scenario Service Provider) on zRule Execution Server for z/OS, you have to add the path of the SSP archive to the zRule Execution Server for z/OS class path.

About this task

To activate Decision Validation Services on zRule Execution Server for z/OS, you must configure it in the instance you create when the initial configuration steps are performed. This involves properly customizing the JCL variables in the **HRBINST** data set member. You must then create a custom SSP that handles the scenarios format that you use, and add this archive to the zRule Execution Server for z/OS class path.

Procedure

1. Make sure that the **HRBINST** variables, **++HBRMODE++**, and **++HBRSSPPORT++**, are properly set.
You must set the **++HBRMODE++** variable to TEST and the **++HBRSSPPORT++** variable to the port number used by the server.
2. In Decision Center, click the **Configure** tab and click **Manage Servers** to specify the path to the server where the SSP has been deployed.

Note: By default, the Rule Execution Server console and the SSP running on zRule Execution Server for z/OS do not have the same port number. You must make sure that you specify the correct port number when you configure the connection to the SSP server. If you want the two port numbers to match, follow the instructions provided in “Repackaging Decision Center to locate the SSP” and “Locating managed XOMs for the SSP in Decision Center” on page 95 before performing this step.

3. Add the custom SSP to the classpath variable of zRule Execution Server for z/OS.

Repackaging Decision Center to locate the SSP:

So that Decision Center can locate the SSP and the Rule Execution Server console on zRule Execution Server for z/OS, you must add an implementation of `ILrSSPResolver` to the Decision Center EAR so that the port number is shared by both of these applications.

About this task

By default, the Rule Execution Server console and the SSP on zRule Execution Server for z/OS do not use the same port number. If you want to use the same port for both the console and the SSP server, you have to create and deploy a class to resolve the location of the SSP in Decision Center.

Important: This action is important if you use the managed XOM feature to store the Java objects for the rule project to test.

Procedure

1. Open Rule Designer and switch to the Java perspective.
2. In the Package Explorer view, click **New > Java Project**.
3. Click **Next** and select the **Libraries** tab.
4. Click **Add Variable ...**
5. In the dialog that displays, select IBM_DC_HOME and click **Extend**.
6. Expand **teamserver > lib**, select `jrules-teamserver.jar`, click **OK** and then **Finish**.
7. In the Package Explorer view, right-click the source folder and click **New > Class**.
8. In the dialog that displays, add the package and the class name, declare the `IlrSSPResolver` as the interface, click **OK** and then **Finish**.
A default class template is generated.
9. Modify the class to map the URL used for the Rule Execution Server console to the URL of the server where the SSP is deployed.
10. Compile the class and package it with the Decision Center EAR, as described below:
 - a. In the Package Explorer view, right-click the Java project and choose **Export**.
 - b. Click **Java > JAR file**.
 - c. Keep the default value, give a name to the JAR file and click **Finish**. The jar file is generated in your workspace.
 - d. In a Command Prompt window, go to the directory `IBM_DC_HOME/bin` and run the command `ant repackage-ear`.
11. Redeploy the repackaged EAR to your application server.
12. In Decision Center, click the **Configure** tab then click **Manage Servers** to specify the path to the server where the SSP is deployed.

Locating managed XOMs for the SSP in Decision Center:

If you do not package the execution object model (XOM) inside the Scenario Service Provider (SSP) on zRule Execution Server for z/OS, you must add the path to the managed XOMs to the rule project that you want to test in Decision Center.

About this task

In Decision Center, you can set the location of the managed XOMs and specify the path of the Rule Execution Server persistence using a descriptor file inside the **Resources** folder of the rule project you want to test.

Procedure

1. Sign in to Decision Center using the administrator user name.
2. In the **Home** tab, select the rule project that you want to test.
3. In the **Explore** tab, click the icon to add a Smart view for Resources.
4. In the **Smart View** wizard:
 - a. Click **Create a new smart folder**.
 - b. Type Resources as the name.
 - c. Select **Find all resources** as a query.
 - d. Select Folder **Displayed Properties** and click **Finish**.
5. In the **Explore** tab, an additional node displays in the panel tree.

6. On the **Resources** node, click the icon that displays.
7. Add a folder and name it META-INF.
8. Click **New** and add the descriptor that defines the managed XOMs and the Rule Execution Server console URL. Name the file deployment.xml. The following file provides an example:

```
<?xml version="1.0" encoding="UTF-8"?><project-deployment
xmlns="http://www.ibm.com/rules/ruleproject/deployment"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<config serverURL="<hostname>:++HBRSSPORT++/res">
<xom>
<uri value="resuri://my-xom.zip/1.0"/>
<uri value="resuri://my-xom-xmarshaller.jar/1.0"/>
</xom>
</config>
</project-deployment>
```

The managed XOM is now attached to the ruleset for the target server.

Verifying your configuration of Decision Center:

You can verify that you have correctly configured Decision Center by publishing some projects, opening the consoles, and running the diagnostics.

Publishing a project to Decision Center:

After completing the configuration, Decision Center is ready to be used but does not contain any rule projects. You publish rule projects from Rule Designer.

About this task

To publish a rule project to Decision Center, the project must be imported into your workspace.

The procedure uses the Decision Center tutorial projects as an example of how to import and publish a rule project. If you want to carry out the Decision Center tutorials, you have to publish the following projects:

- loanvalidation-rules (with loanvalidation-xom)
- loanvalidation-rules-dependent
- squery-loanvalidation-rules (with squery-loanvalidation-xom)

Procedure

1. To open Rule Designer, click **Start > All Programs > IBM > package_group > Rule Designer**.
2. In Rule Designer click **File > Import > General > Existing Projects into Workspace**, and click **Next**.
3. Click **Select root directory**, browse to <InstallDir>/studio/tutorials/shared, and click **OK**.
4. Select the projects and click **Finish**.
5. Right-click the loanvalidation-rules rule project, and click **Decision Center > Connect**.
6. Complete the Decision Center Configuration dialog as follows. The warning message Connection not established displays until you establish the connection.

User name

rtsAdmin

Password

rtsAdmin

URL http://localhost:<port>/teamserver

Data source

Leave this field empty.

Note: If security is enabled, use https://localhost:<PORT_NUMBER>/teamserver

7. Click **Connect**.

The connection is established when the warning message closes and the **Project configuration** area becomes active.

8. In the **Project configuration** area, check that **Create a new project on Decision Center** is selected, and then click **Finish**.

9. The Synchronize Complete - Decision Center Participant dialog opens when the publishing process is complete. Click **OK** to close this dialog.

10. A dialog opens asking you if you want to change to Team Synchronizing perspective. Click **Yes**.

An empty Synchronize view opens, indicating that the projects in Rule Designer and Decision Center are the same. This means that your rules are now published to Decision Center.

11. Repeat for the other rule projects.

What to do next

You can now open the Decision Center Enterprise console and perform diagnostics.

Opening the Decision Center consoles:

After you have deployed the Decision Center EAR or WAR to your application server, you can open the Decision Center consoles.

You can open the consoles by using the following URLs in a web browser:

- **Enterprise console:** http://localhost:<PORT_NUMBER>/teamserver
- **Business console:** http://localhost:<PORT_NUMBER>/decisioncenter

Note: If your browser is not running on the same host as the application server, replace localhost with the address of the machine. If your web application is mapped to a host with a port that is different from the default port, use the port number of the host.

By default, the data source is jdbc/ilogDataSource. If you want to specify a different data source, you have to pass it as a request parameter in the URL. For example:

http://localhost:7001/teamserver?datasource=jdbc/serverextendedbrm.

The locale of the sign-in page is English by default. You can specify a locale parameter in the URL that switches the sign-in page to the required locale. For example:

http://localhost:<port>/teamserver?locale=es (assuming that your message files are localized).

If you sign in with another locale in the URL and want to change the locale afterward, click **Options** in the top banner of the Enterprise console or **Profile** in the Business console. This saves the locale and restores it the next time you sign in.

If you open Decision Center but no database exists, you automatically access the Installation Settings wizard with only the **Install** tab available.

After completing the installation, Decision Center is ready to use but does not contain a rule project. You have to publish a rule project from Rule Designer.

A diagnostics tool, available in the Configure tab of the Enterprise Console, shows a report on the status of your Decision Center configuration.

To learn more about Decision Center, see Decision Center.

Configuring topology 5: batch execution and an embedded zRule Execution Server for z/OS instance

This topology consists of a zRule Execution Server for z/OS instance that is run locally in the batch address space. You must configure the batch environment to run a batch job.

Configuring the Java batch environment to run a Java batch job

You must configure the Java batch environment to run a Java batch job.

Configuring the Java batch environment:

The `ra.xml` file must be configured to provide configuration information for a Java batch program. You can use the `HBRCJCFG` job to configure the file and the Java batch environment.

Before you begin

You must use control statements to create and configure the execution environment. For more information, see “Customizing topology 5: batch execution and an embedded zRule Execution Server for z/OS instance” on page 13.

About this task

By submitting `HBRCJCFG`, the following processes are completed automatically.

1. `HBRCJCFG` executes the `RESSetupZ` Java class.
2. The `RESSetupZ` Java class reads properties from certain members inside of `++HBRWORKDS++.SHBRP`, as specified in the `HBRENVPR` DD statement in `HBRCJCFG`.
3. `RESSetupZ` parses these properties and generates the `ra.xml` file.

`HBRCJCFG` also creates a `logging.properties` file. This file contains the log level that is derived from the `HBRTTRACELEVEL` property in the `++HBRWORKDS++.SHBRP(HBRCMMN)` member.

Procedure

1. Optional: Set `HBRC_CONSOLECOM=NO` in `++HBRWORKDS++.SHBRP(HBRCMMN)` to disable the connection to the Rule Execution Server console.

- If you do not want the ruleset to be updated while the batch job is running, or you do not want to collect and display ruleset statistics, the connection to the Rule Execution Server console might not be required for management purposes.
2. Submit the HBRCJCFG job.

Results

The ra.xml file is in ++HBRWORKPATH++/config/res when it is successfully generated.

Running the sample Java batch program:

The HBRMINBJ sample job is used to launch a JVM to run the MiniLoanDemo Java sample.

About this task

By submitting HBRMINBJ, the following processes are completed automatically.

1. HBRMINBJ adds ++HBRWORKPATH++/config/res to the class path. This directory contains the generated ra.xml file. Other JAR files that are required for execution are also included.
2. The MiniLoanDemo Java sample sets up a Rule Execution Server by using a IlrJ2SESessionFactory, and then executes rules by using the input data.
3. The member that contains the input data ++HBRWORKDS++.SHBRPARM(HBRSCEN) is specified in the SCENARIO DD statement within HBRCJCFG.

Note: HBRMINBJ executes the MiniLoanDemo Java sample by using BPXBATSL, so that it launches in the same address space, and can therefore read the SCENARIO DD statement.

Procedure

Submit HBRMINBJ to run the MiniLoanDemo Java sample.

Step 9: Verifying your rules execution environment configuration

Verify that you have successfully configured your new instance or server group.

About this task

Use diagnostic tests to verify that you have successfully configured your new instance or server group. When you run diagnostics for servers that are in a cluster, the local XU connectivity check for the Rule Execution Server console returns the message: Warning JNDI lookup to retrieve local XU failed. This is to be expected because the XUs are all located on other servers.

Procedure

1. Display the Rule Execution Server console by opening a browser and navigating to `http://<hostname>:++HBRCONSOLEPORT++/res`, where *hostname* is the TCP/IP name of the z/OS machine where you are running the server and ++HBRCONSOLEPORT++ is one of the following:
 - For zRule Execution Server for z/OS, the HTTP port used by the console to communicate with the server.
 - For WebSphere Application Server, the port number on the WebSphere Application Server installation.

2. Sign in to the console using a user ID in the resAdministrators group. If you cannot sign in, make sure you have the required authorization. For more information, see Troubleshooting.
3. For the new server instance or for each server in the server group:
 - a. Click the **Diagnostics** tab.
 - b. Click **Run Diagnostics**. A report lists the diagnostic tests that ran.
 - c. Click **Expand All** to view the details.

Configuring a COBOL rule subprogram into a COBOL application

You can integrate the generated COBOL code into a COBOL calling program using a static or a dynamic link, or using CICS channels and containers.

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features](#) for migration details.

COBOL-generated code integration

You can generate a COBOL program to execute rules and then link it into a COBOL calling application.

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features](#) for migration details.

Generate the COBOL program in Rule Designer, Decision Center, or using an Ant command. Then integrate the generated COBOL program with a COBOL calling program and run both on a z/OS subsystem.

You can run the generated COBOL program in the following types of application:

- Batch
- CICS
- IMS

If you generate the code in Rule Designer, the source file resides in the directory you specified during code generation, ready to upload to z/OS. If you use Decision Center, you download the output from the Decision Center code generation process onto your local machine. If you used an Ant command, the generated COBOL program is saved to: <InstallDir>/studio/lib.

Linking code by using a COBOL static link

You can link the generated COBOL program by adding a static link to each program that calls it.

About this task

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features](#) for migration details.

Using a static link you can compile, link, and then run the generated COBOL code inside the calling program, but you must statically link it with every program that calls it. If the generated program changes, you must relink it to all its calling programs for the changes to take effect.

You link the generated COBOL program into the calling program using a CALL statement.

Procedure

1. Declare the generated COBOL program as a constant in the WORKING-STORAGE section of the calling program. For example:

```
01 GENERATED-PROG PIC X(8) VALUE 'MINICBL'
```

Where MINICBL is the name of the generated COBOL program file.
2. Add a CALL statement to the PROCEDURE DIVISION of the calling program that uses this constant. For example:

```
CALL GENERATED-PROG USING BORROWER LOAN
```

You reference each Top Level data item with a USING clause.
3. Compile the calling program.

Results

The rules are now ready to be executed.

Example static link to a batch program

The following code extract shows how to link the generated COBOL code to a COBOL calling program, using a CALL statement in the PROCEDURE DIVISION. The XOM is based on a COBOL copybook named `miniloan` that has two Top Level data items: LOAN and BORROWER. These items translate into two classes in the BOM, named Loan and Borrower. The generated COBOL program file is named MINICBL.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. MAIN.  
  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
COPY MINILOAN.  
01 TABLEINDEX PIC 999.  
01 GENERATED-PROGRAM PIC X(8) VALUE "MINICBL".  
  
PROCEDURE DIVISION.  
ROOT.  
MOVE "JOHN" TO NAME OF BORROWER  
MOVE 100 TO CREDITSCORE OF BORROWER  
MOVE 10000 TO YEARLYINCOME OF BORROWER  
MOVE 10000 TO AMOUNT OF LOAN  
MOVE 5 TO YEARLYINTERESTRATE OF LOAN
```

```

MOVE 1000 TO YEARLYREPAYMENT OF LOAN
MOVE "N" TO APPROVED OF LOAN
MOVE 0 TO MESSAGECOUNT OF LOAN

CALL GENERATED-PROGRAM USING LOAN BORROWER

IF APPROVED OF LOAN = "Y"
  DISPLAY "Loan approved"
ELSE
  DISPLAY "Loan not approved"
  MOVE 1 TO TABLEINDEX
  PERFORM UNTIL TABLEINDEX > MESSAGECOUNT OF LOAN
    DISPLAY MESSAGES OF LOAN (TABLEINDEX)
    COMPUTE TABLEINDEX = TABLEINDEX + 1
  END-PERFORM
END-IF
GOBACK.

```

Linking code by using a COBOL dynamic link

Use a dynamic link to link the generated COBOL program independently from the calling program.

About this task

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features for migration details](#).

Using a dynamic link, you can link the generated COBOL program independently from the calling program. This way of linking is similar to using a CICS EXEC CICS LINK, except that with a dynamic link, both the generated COBOL program and the COBOL calling program must reside in the same CICS system.

Tip: With an EXEC CICS LINK, the programs can be in different CICS systems.

Procedure

1. Declare the generated COBOL program as a constant in the WORKING-STORAGE section of the calling program. For example:


```
01 GENERATED-PROG PIC X(8) VALUE 'MINICBL'
```

 Where MINICBL is the name of the generated COBOL program file.
2. Add a CALL statement to the PROCEDURE DIVISION of the calling program that uses this constant. For example:


```
CALL GENERATED-PROG USING BORROWER LOAN
```

 You reference each Top Level data item with a USING clause.
3. To link the COBOL program dynamically, select one of the following compile options:
 - For a batch program, specify DYNAM=YES.
 - For a CICS program, specify DYNAM=NO.
4. To link the COBOL code into a CICS program, specify CICS=NO.

Note: If you specify CICS=YES, CICS expects a COMMAREA to be passed to the COBOL-generated program. The generated COBOL program does not include a dummy COMMAREA, so specifying CICS=YES can result in 0C4 ABENDs occurring.

Results

When you run the calling program, the rules are executed.

Calling a CICS wrapper program to integrate the code

If you use the CICS usage option when you generate the COBOL code, you must also update the CICS calling program so that it calls the wrapper program to execute the COBOL program for rule execution.

About this task

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features](#) for migration details.

If you want to host the generated COBOL program in a different CICS region to the calling application, you can use the CICS usage option. In this case, the COBOL code generator generates a CICS wrapper program, as well as the COBOL code for rule execution. If you use the CICS usage option, you must also update the CICS calling program so that it calls the wrapper program to execute the COBOL program for rule execution.

Procedure

To update the CICS calling program:

1. Add an EXEC CICS PUT CONTAINER statement for each group item.
For example, if you have two group items, LOAN and BORROWER, you might create the following statements:

```
EXEC CICS PUT CONTAINER(LOAN-CONTAINER)
          CHANNEL(MINILOAN-CHANNEL)
          FROM(Loan) FLENGTH(Length of Loan)
          END-EXEC
```

```
EXEC CICS PUT CONTAINER(BORROWER-CONTAINER)
          CHANNEL(MINILOAN-CHANNEL)
          FROM(Borrower) FLENGTH(Length of Borrower)
          END-EXEC
```

2. Replace the COBOL call statement with an EXEC CICS LINK PROGRAM statement that specifies the name of the channel that you want to pass to the wrapper program.
3. Add an EXEC CICS GET CONTAINER statement for each Top Level data item modified by the generated COBOL program.

You must have the **GET CONTAINER** commands to get any information from the containers that has been modified by the COBOL program. The **GET CONTAINER** command picks up the information from the **PUT CONTAINER** command at the end of the wrapper program.

Note: If the generated COBOL program increases the size of a data item, for example, if the generated program modifies an OCCURS DEPENDING ON variable, use COBOL pointers to retrieve the data.

Results

You can now translate, compile and link the calling program, the wrapper program, and the generated COBOL code. This action is slightly different from a normal COBOL compile because all the EXEC CICS statements must be translated to compile the code successfully.

Configuring your system to collect usage data

If your IBM license requires you to monitor your usage of COBOL programs generated by COBOL management, you must configure your system depending on your execution environment.

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features](#) for migration details.

Configuring zRule Execution Server for z/OS to collect usage data

You can collect usage data on an instance of zRule Execution Server for z/OS.

About this task

To activate the collection of usage data on an instance of zRule Execution Server for z/OS, add a variable to the ++HBRWORKDS+.SHBRPARM data set.

Procedure

- For an instance of zRule Execution Server for z/OS server, add the following variable to the HBRMSTR member:
HBRIFAUSAGE=YES
- For an instance of zRule Execution Server for z/OS on CICS JVM server, add the following variable to the HBRCICSJ member:
HBRIFAUSAGE=YES

Configuring Rule Execution Server on WebSphere Application Server for z/OS to collect data

You can collect usage data on an instance of Rule Execution Server on WebSphere Application Server for z/OS.

About this task

Activate the collection of usage data from an instance of Rule Execution Server on WebSphere Application Server for z/OS by submitting a JCL job.

Procedure

Submit one of the following jobs depending on your installation:

- SHBRJCL(HBRDMREG): if your installed product includes rules and events components
- SHBRJCL(HBRBRREG): if your installed product includes the rules components only

Configuring COBOL rule subprograms to collect data

To configure usage data collection for a COBOL rule subprogram you must carry out a number of steps to use the System Management Facility (SMF).

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See *Deprecated features* for migration details.

Usage monitoring tools

The System Management Facility (SMF) collects runtime usage data for COBOL source programs generated by COBOL management at runtime. The generated rule execution program communicates with SMF through the COBOL Resource Manager.

Note:



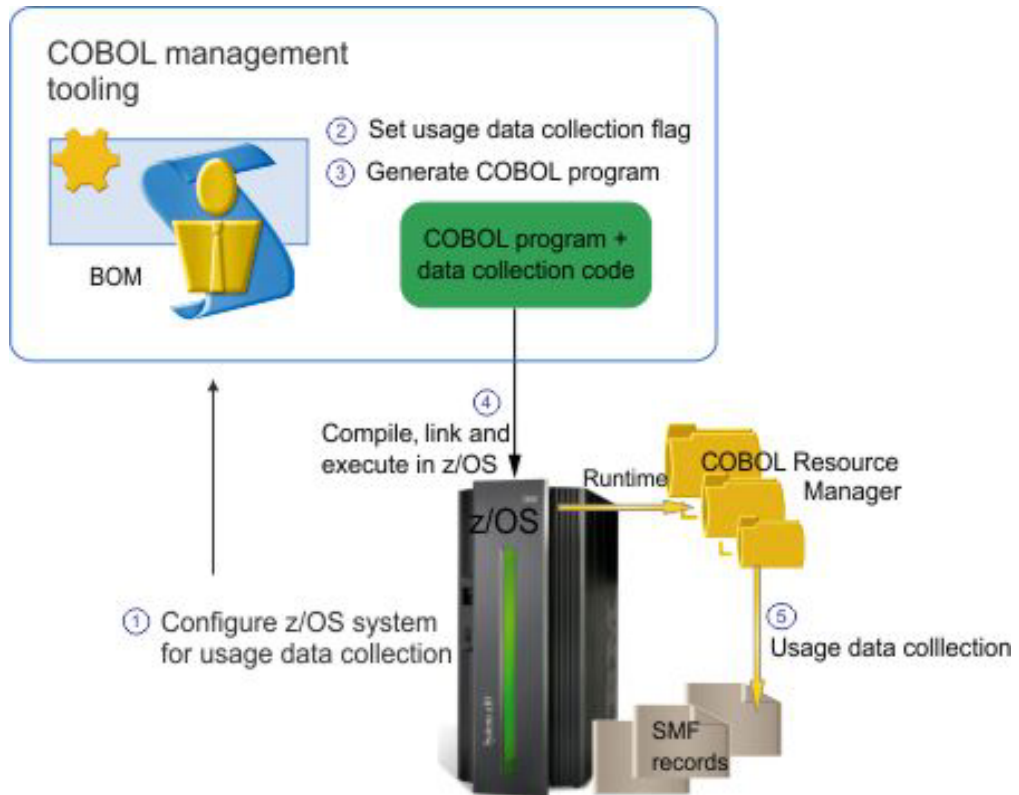
Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See *Deprecated features* for migration details.

If your IBM license requires you to monitor your usage of the COBOL source programs that you generate using COBOL, you must set the usage data collection flag in Rule Designer so that you can collect COBOL source program usage data. Usage data is collected at runtime by the z/OS[®] System Management Facility (SMF).

Communication between generated COBOL source program and SMF is handled by the COBOL Resource Manager, which executes as a z/OS subsystem. At runtime, the generated COBOL source program notifies the Resource Manager when execution starts and ends. The Resource Manager passes this data to SMF, and SMF generates the appropriate program usage records.

To enable usage data collection, you must configure z/OS, set the usage data collection flag in Rule Designer before you generate the COBOL code, and then carry out some additional tasks when compiling and linking the generated program.

The following diagram illustrates the role of the Resource Manager, and the overall sequence of events that result in usage data collection.



Configuring z/OS

For the generated COBOL source programs to participate in program usage data collection, you must carry out a number of z/OS configuration tasks.

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features](#) for migration details.

Configuration tasks:

To collect usage data, you must configure the required libraries and z/OS components, set the usage data collection flag, and then carry out some additional tasks when compiling and linking the generated code.

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features](#) for migration details.

Before you can collect usage data for generated COBOL source programs, you must carry out the following z/OS configuration tasks:

1. Define the COBOL Resource Manager as a z/OS subsystem.
2. Configure the Resource Manager subsystem

Note:

These are one-off tasks that configure your system for program usage monitoring for any generated COBOL programs that you run on it.

Defining the Resource Manager as a z/OS subsystem:

You must define the COBOL Resource Manager as a valid z/OS subsystem so that the z/OS system recognizes it.

About this task

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See *Deprecated features* for migration details.

You define the Resource Manager as a z/OS subsystem by giving it a name and then adding it to the list of defined z/OS subsystems. You can define it dynamically, or you can define it permanently by adding a definition to the system parameter library (parmlib) member IEFSSNxx.

Procedure

1. Do one of the following, depending on whether you want to add the subsystem dynamically or permanently:

Add subsystem dynamically	Add subsystem permanently
Issue the following z/OS system command: SETSSI ADD,SUBNAME=++SUBS++	Add the following definition to the IEFSSNxx member: SUBSYS SUBNAME(++SUBS++)

Where ++SUBS++ is the name given to the COBOL Resource Manager, specific to your installation and naming conventions.

2. Optional: Check that the Resource Manager has been added to the list of z/OS subsystems using the following z/OS system command:

D SSI,LIST,SUB=++SUBS++

Loading the COBOL load module into the LPA:

The BRPF7100 program is the COBOL load module that interacts with SMF. You must load this program into the link pack area (LPA). You can load the program dynamically, or load it permanently by adding a definition to the parmlib member PROGxx.

About this task

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See *Deprecated features* for migration details.

To load the program use one of the following methods:

Procedure

- SETPROG LPA,ADD,MODNAME=(BRCF7100),DSNAME=++HBRHLQ++.SHBRAUTH where ++HBRHLQ++ is your user-defined high level qualifier. This command loads the module dynamically.
- LPA ADD MODNAME(BRCF7100) DSNAME(++HBRHLQ++.SHBRAUTH) Add the definition to the parmlib member PROGxx.
- D PROG,LPA,MODNAME=BRCF7100 This z/OS system command checks that the program has been loaded into the LPA.

Configuring the Resource Manager subsystem:

Before you can start the COBOL Resource Manager, you must configure the JCL member BRCSUBS.

About this task

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See *Deprecated features* for migration details.

To be able to start the COBOL Resource Manager, you configure the JCL member BRCSUBS, which is stored in the SHBRJCL library. The JCL member tells you which parts of the JCL you must change. You can submit it as a job or as a started task.

Procedure

To customize the JCL:

1. Add your installation-specific jobcard.
2. Update the STEPLIB.
3. Replace the ++SUBS++ variable with the subsystem name defined for the COBOL Resource Manager.
This definition tells the generated COBOL program which subsystem to connect to.
4. Submit the job control.

Note:

- This is a long-running job and must be executed while the generated COBOL program or programs are executing.
- To terminate the COBOL Resource Manager, issue the z/OS command **STOP <jobname>**.

Results

You now set the usage monitoring flag in Rule Designer. See Monitoring generated COBOL program usage in the distributed platform part of this information center.

Compiling, linking, and running your programs

To collect program usage data, you perform some additional steps when compiling, linking, and running the generated COBOL program.

About this task

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See Deprecated features for migration details.

Before you execute the generated COBOL program, make sure that you have carried out the following tasks when compiling, linking, and running the generated COBOL program.

Procedure

To compile, link and run your programs:

1. To enable the compiler to retrieve the COBOL copybooks provided for usage data collection, add the COBOL copy library, `++HBRHLQ++.SHBRCOBC`, to the compile step SYSLIB DD statement.
For example: `//SYSLIB DD DISP=SHR,DSN=ZILOG ++HBRHLQ++.SHBRCOBC` You might already have several libraries specified to SYSLIB. Add SHBRCOBC to the concatenation.
2. To link your generated COBOL programs correctly:
 - a. Add a reference to the COBOL load library in the linkedit step
`//SHBRLOAD DD DISP=SHR,DSN=++HBRHLQ++.SHBRLOAD`
You add this reference so that the linkedit step can resolve the COBOL Resource Manager BRCC stubs.
 - b. Include the following binder statement in the linkedit step, to make sure that the binder includes the stub module that COBOL management supplies.
`INCLUDE SHBRLOAD(BRCCSTUB)`
3. To enable the generated COBOL program to identify the Resource Manager to use when collecting usage monitoring data, add the following DD statement to the job step that executes your COBOL program:
`//R4CS++SUBS++ DD DUMMY`

Where ++SUBS++ is the subsystem name, for example: //R4CSBRA2 DD DUMMY
At run time, the generated COBOL program must be able to identify which Resource Manager to use usage data collection.

Sample COBOL code for usage data collection

When you enable usage data collection, the generated COBOL source program adds a reference to the BRCCONST and BRCCONWS copybooks.

Note:



Deprecated feature: The COBOL code generation feature is deprecated in V8.5. This feature will be removed in a future release of the product. Use the COBOL Generation Project Migration wizard to migrate your Rules for COBOL project to a zRule Execution Server for z/OS compatible rule project. See [Deprecated features for migration details](#).

The following code extract shows an example of the type of code that is generated when you set the usage data collection flag. It shows the references to the BRCCONST and BRCCONWS copybooks in the WORKING-STORAGE SECTION, and the code that handles usage data collection in the PROCEDURE DIVISION.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. R4CZOSRM IS INITIAL PROGRAM.  
  
* This COBOL program is generated from the following ruleset: miniloan  
* Created at 2010/03/05 10:07:02.  
  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
COPY BRCCONST.  
COPY BRCCONWS.  
  
.  
.  
.  
  
LINKAGE SECTION.  
COPY MINILOAN.  
  
PROCEDURE DIVISION USING BORROWER LOAN.  
TASK1-SMF.  
INITIALIZE BRCC-CONN-AREA  
MOVE "miniloan" TO BRCC-CONN-RULESET-NAME OF BRCC-R4C-DATA  
MOVE "20100305" TO BRCC-CONN-CREATE-DATE OF BRCC-R4C-DATA  
MOVE "100702" TO BRCC-CONN-CREATE-TIME OF BRCC-R4C-DATA  
MOVE BRCC-CONN-RS OF BRCC-R4C-CONSTANTS TO  
BRCC-CONN-CREATE-METHOD OF BRCC-R4C-DATA  
COPY BRCCINIT.  
PERFORM TASK2-MAINFLOW  
COPY BRCCTERM.  
GOBACK.  
  
* RuleFlow mainflow  
TASK2-MAINFLOW.  
PERFORM TASK3-MAINFLOW-VALIDATION  
PERFORM TASK4-MAINFLOW-ELIGIBILITY.  
* RuleTask mainflow#validation  
  
.  
.  
.
```

Configuring your system to collect execution data

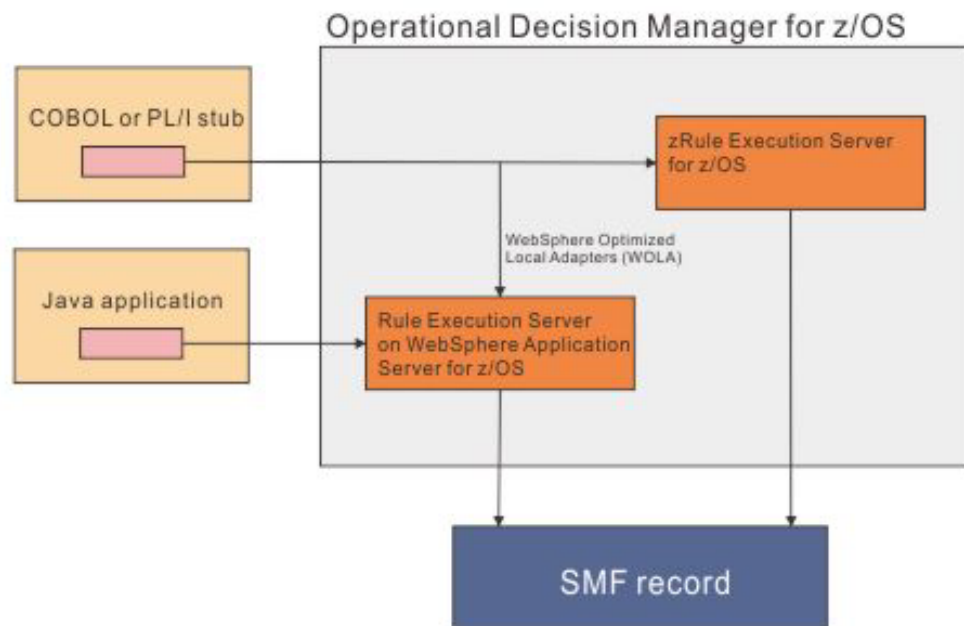
If your IBM license requires you to monitor the number of decisions that are executed on an LPAR, you must configure your system depending on your execution environment.

Collection of execution data by using SMF type 120 subtype 100 records

The System Management Facility (SMF) collects execution data such as the number of decisions that are executed by your application at run time.

The SMF records are written by the rule execution environment, which can be zRule Execution Server for z/OS, a CICS rule-owning region, or Rule Execution Server on WebSphere Application Server for z/OS.

The following diagram shows the sequence of data that results in execution data collection.



The SMF type 120 subtype 100 records are written to SMF by using the global SMF properties INTVAL and SYNCVAL. If these values are changed while the execution environment is active, it does not use the new values immediately. Instead, the new values will be adopted when the next SMF record is written by the execution environment.

The execution data collection function does not currently support the use of SUBSYS with the value STC or the field INTERVAL in the SMF parameters.

Execution data that is collected in the SMF records

The execution data, such as the number of decisions that are executed at run time, is written to System Management Facility (SMF) records.

The following tables show the information that is written to the SMF type 120 subtype 100 records. Each record contains the count of the rulesets that are executed in an interval by an execution unit (XU).

An SMF formatter for basic record formatting is provided as a source code sample.

Table 6. SMFRecordHeader

Field name	Data type	Description
SMF120FLG	char	System indicator
SMF120RTY	char	Record type 120 (x'78')
SMF120TME	char[4]	Time when SMF moved record
SMF120DTE	char[4]	Date when SMF moved record
SMF120SID	char[4]	System ID
SMF120SSI	char[4]	Not used.
SMF120STY	uint16_t	Record subtype
SMF120HDV	uint16_t	ODM record version number
SMF120HDO	uint32_t	Header that is offset in a record
SMF120HDL	uint32_t	Header length
SMF120HDN	uint32_t	Number of headers

Table 7. HBRSMF120ST100RecordHeader

Field name	Data type	Description
SMF120VER	char[16]	ODM version Example: 8.6.0.0
SMF120XUL	char[16]	XU location Examples: HBR1, IYBNCAN
SMF120XUT	char[32]	XU location type Examples: zRule Execution Server for z/OS, CICS
SMF120SDT	char[16]	SMF interval start date
SMF120STM	char[16]	SMF interval start time
SMF120EDT	char[16]	SMF interval end date
SMF120ETM	char[16]	SMF interval end time
SMF120EXO	uint32_t	Exec section start point
SMF120EXL	uint32_t	Length of the whole exec section
SMF120EXN	uint32_t	Total number of exec segments

Table 8. HBRSMF120ST100RecordExec

Field name	Data type	Description
RULEXNUM	uint32_t	Ruleset successful execution count
RULEXBAD	uint32_t	Ruleset failed execution count

Table 8. HBRSMF120ST100RecordExec (continued)

Field name	Data type	Description
RULEXFSUM	uint32_t	Ruleset sum of fired rules
RULEXPATH	char[256]	Ruleset execution path

Related concepts:

“Sample code to export execution data in CSV format” on page 116

To collect the System Management Facility (SMF) type 120 subtype 100 records and export them in a comma-separated value (CSV) format, you must edit and submit ++HBRWORKDS++.SHBRJCL(HBRSMFC).

Related information:

“Sample code to format execution data” on page 115

To collect and view the System Management Facility (SMF) type 120 subtype 100 records, you must edit and submit ++HBRWORKDS++.SHBRJCL(HBRSMFP).

Configuring zRule Execution Server for z/OS to collect execution data

You must be authorized to the BPX.SMF profile of the FACILITY class to write System Management Facility (SMF) records. You can enable the SMF recording feature to collect execution data by setting a property.

About this task

Authorize the user and the zRule Execution Server for z/OS to be able to write SMF records.

Procedure

1. Ensure that the user that runs zRule Execution Server for z/OS is authorized to the BPX.SMF profile of the FACILITY class. Use the following commands:

```
PERMIT BPX.SMF CLASS(FACILITY) ID(<ZRESTCID>) ACCESS(READ)
```

```
SETROPTS RACLIST(FACILITY) REFRESH
```

2. Set the following property in the ++HBRWORKDS++.SHBRPARM(HBRMSTR) member of the zRule Execution Server for z/OS which requires the SMF execution data:

```
HBRSMFST100=YES
```

Configuring a CICS rule-owning region to collect execution data

You must be authorized to the BPX.SMF profile of the FACILITY class to write System Management Facility (SMF) records. You can enable the SMF recording feature in a CICS rule-owning region to collect execution data by setting a property.

About this task

Authorize the user and the zRule Execution Server for z/OS in the CICS JVM Server to be able to write SMF records.

Note: The CICS rule-owning region begins reporting execution data when the HBRM transaction is run and the zRule Execution Server for z/OS starts inside the CICS JVM server. If the HBRD transaction is run instead, the CICS rule-owning region stops reporting execution data and SMF records are not written.

Procedure

1. Ensure that the user that runs CICS is authorized to the BPX.SMF profile of the FACILITY class. Use the following commands:
PERMIT BPX.SMF CLASS(FACILITY) ID(<CICSSTCID>) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
2. Set the following property in the ++HBRWORKDS++.SHBRPARAM(HBRCICSJ) member of the CICS rule-owning region that requires the SMF execution data:
HBRSMFST100=YES

Configuring Rule Execution Server on WebSphere Application Server for z/OS to collect execution data

You must be authorized to the BPX.SMF profile of the FACILITY class to write System Management Facility (SMF) records. You must configure a few things in Rule Execution Server on WebSphere Application Server for z/OS before you start collecting execution data.

Procedure

1. Ensure that the user that runs WebSphere Application Server is authorized to the BPX.SMF profile of the FACILITY class. Use the following commands:
PERMIT BPX.SMF CLASS(FACILITY) ID(<WASSTCID>) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
2. Add the SMFPlugin to the plug-ins list for the execution unit (XU).
 - a. Go to **Resource > Resource Adapters** . Click **Resource adapters**, and then click **J2C connection factories > xu_cf**.
 - b. Add `<InstallDir>/zexecutionserver/lib/hbrsmf.jar` to the class path.
 - c. Under **Additional Properties**, click **Custom properties**.
 - d. Click the property with the name `plugins`.
 - e. Add the following string to the `plugins` property:
`{pluginClass=com.ibm.rules.hbr.smf.SMFPlugin},`

Note: You must have a comma between two entries to separate one plug-in from the next plug-in.

3. Add the SMF function to the native library path.

Note: If you are using WebSphere Application Server Network Deployment, the following steps from 3-c to 3-f must be repeated for every server that you require the SMF data collection for.

- a. Open WebSphere Integrated Solutions Console.
- b. In the side panel, open **Servers > Server types > WebSphere application servers**.
- c. On the Application servers page, click the name of your server.
- d. Under **Server Infrastructure**, expand **Java and Process Management** and click **Process definition**.

In WebSphere Application Server for z/OS, an additional layer provides three resources that can be administered: **Adjunct**, **Control**, and **Servant**. If you are working in that environment, select **Servant**.

- e. Under **Additional Properties**, click **Environment Entries**.
- f. Click **New**. Add an entry with the name `LIBPATH` and the value `<InstallDir>/lib`.

- g. Click **OK**, and then click **Save** to save the changes directly to the master configuration.
4. Restart your server to begin collecting execution data records.

Configuring Java batch to collect execution data

The owner of the Java batch job must be authorized to the BPX.SMF profile of the FACILITY class to write System Management Facility (SMF) records. You can enable the SMF recording feature to collect execution data by setting a property.

About this task

Authorize the user who is executing the Java batch job to be able to write SMF records.

Procedure

1. Ensure that the user who runs the Java batch job is authorized to the BPX.SMF profile of the FACILITY class. Use the following commands:


```
PERMIT BPX.SMF CLASS(FACILITY) ID(<JAVABATCHID>)ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```
2. Set the following property in the ++HBRWORKDS++.SHBRPARM(HBRBATCH) member of the Java batch job that requires the SMF execution data:


```
HBRSMFST100=YES
```
3. Run the Java batch configuration JCL to regenerate the ra.xml configuration file. Submit ++HBRWORKDS++(HBRJCFCG).
4. Ensure that the JVM that starts the Java batch workload has the system property set hbr.javabatch="true". This system property set can be passed in via the Java command line by using the following technique:


```
-Dhbr.javabatch="true"
```

Results

SMF type 120 subtype 100 records is now written by the Java batch workload. The following fields are set when the workload is a Java batch:

- SMF120XUL contains the first 16 characters of the main class name which Java is executing. For example, running the class `com.ibm.rules.hbr.samples.miniloan.MiniLoanDemo` results in the value `MiniLoanDemo` being shown in SMF120XUL.
- SMF120XUT contains the string `Embedded Rule Execution Server`.

Sample code to format execution data

To collect and view the System Management Facility (SMF) type 120 subtype 100 records, you must edit and submit ++HBRWORKDS++.SHBRJCL(HBRSMFP).

Before you submit ++HBRWORKDS++.SHBRJCL(HBRSMFP), ensure that the values for the DD statements DUMPINA and DUMPINB are updated to reflect the SMF data sets for your system.

When the execution completes, you see that the SMF type 120 subtype 100 records are in a readable format, as in the following example output:

```
*****
* SMFRecordHeader *
*****
SMF120RTY = 120
```

```

SMF120SID = MVGA
SMF120STY = 100
SMF120HDV = 1
SMF120HDO = 36
SMF120HDL = 140
SMF120HDN = 1
*****
* HBRSMF120ST100RecordHeader *
*****
SMF120VER = 8.6.0.0
SMF120XUL = BRAV
SMF120XUT = zRule Execution Server
SMF120SDT = 9/30/14
SMF120STM = 8:45:17 AM
SMF120EDT = 9/30/14
SMF120ETM = 8:50:00 AM
SMF120EXO = 172
SMF120EXL = 536
SMF120EXN = 2
*****
* HBRSMF120ST100RecordExec *
*****
RULEXNUM = 6
RULEXBAD = 0
RULEXFSUM = 5
RULEXPATH = /MiniLoanDemoRuleApp/1.0/MiniLoanDemo/1.0
*****
* HBRSMF120ST100RecordExec *
*****
RULEXNUM = 6
RULEXBAD = 0
RULEXFSUM = 5
RULEXPATH = /MiniLoanDemoPLIRuleApp/1.0/MiniLoanDemoPLI/1.0

```

The code for the sample HBRSMFP program is in ++HBRHLQ++.SHBRXLCS and the required header files are in ++HBRHLQ++.SHBRXLCH.

Related information:

“Execution data that is collected in the SMF records” on page 111

The execution data, such as the number of decisions that are executed at run time, is written to System Management Facility (SMF) records.

Sample code to export execution data in CSV format

To collect the System Management Facility (SMF) type 120 subtype 100 records and export them in a comma-separated value (CSV) format, you must edit and submit ++HBRWORKDS++.SHBRJCL(HBRSMFC).

Before you submit ++HBRWORKDS++.SHBRJCL(HBRSMFC), ensure that the values for the DD statements DUMPINA and DUMPINB are updated to reflect the SMF data sets for your system, and that the DD statement SMFOUT points to a valid writeable USS location for the output CSV file.

When the execution completes, a CSV file is generated with details of the SMF type 120 subtype 100 records.

Related information:

“Execution data that is collected in the SMF records” on page 111

The execution data, such as the number of decisions that are executed at run time, is written to System Management Facility (SMF) records.

z/OS configuration variables

The variables that you use for configuring an Operational Decision Manager for z/OS instance are held in the HBRINST or HBRCTRL data set member. To be valid, the values you specify must not contain spaces.

Overview

Variables for configuring the environment have the following notation: ++VARIABLE_NAME++. The equivalent runtime variables do not have any special notation.

The variables in the ++HBRWORKDS++.SHBRP(HBRINST) data set member must be customized. The ++HBRWORKDS++ variable represents the high-level qualifier that is assigned to the working data sets. The HBRUPTI job must then be submitted to generate the working data sets. The data sets created by the HBRUPTI job are then used to configure the runtime instance.

Important: For some variables, after you run HBRUPTI, you must check the working data set to ensure that the JCL is valid. If the JCL is not valid because, for example, an entry was truncated, you must manually edit the JCL. For more information, see the variables descriptions.

Set the configuration variables to configure a new Operational Decision Manager for z/OS instance. The configuration variables include: Postinstallation configuration variables, and variables that initialize the runtime variables in the other members of this data set, and in other data sets. These variables are also used for generating the working data sets for configuring and running an execution environment. The version of HBRINST or HBRCTRL must match the version of the zRule Execution Server for z/OS instance you configure. The HBRINST or HBRCTRL data set member belongs to SHBRP.

Configuration variables

- Configuration variables for configuring a new Decision Server for z/OS instance.
- Configuration variables for configuring CICS to execute rules on a zRule Execution Server for z/OS instance.
- Configuration variables for configuring zRule Execution Server for z/OS running on a CICS JVM server.
- Configuration variables for configuring IMS to execute rules on a zRule Execution Server for z/OS instance.
- Configuration variables for configuring a DB2 database for Operational Decision Manager for z/OS.
- Configuration variables for configuring WebSphere Application Server for Operational Decision Manager for z/OS.
- Configuration variables for enabling wsadmin scripts for Operational Decision Manager for z/OS.
- Configuration variables for configuring a connection to the Rules for COBOL subsystem.
- Configuration variables for connecting to a Rule Execution Server on WebSphere Application Server through WOLA.

Table 9. Configuration variables for configuring a new Decision Server for z/OS instance.

Variable	Description
++HBRHLQ++	<p>The high-level qualifier for Operational Decision Manager for z/OS data sets.</p> <p>Example: HBR.ODM</p> <p>The value of this variable must be different from the value of the ++HBRWORKDS++ variable.</p> <p>This variable is mandatory.</p>
++HBRINSTPATH++	<p>The installation root directory.</p> <p>Example: /usr/lpp/zDM/V8R6M0</p> <p>This variable is mandatory.</p>
++HBRWORKPATH++	<p>The working directory that is used for the execution environment, unless it is overridden in the control statements in HBRCTRL.</p> <p>Example: /u/HBR1</p> <p>If this variable is 35 characters or less in length, the value can be used by all working data sets. If the value exceeds 35 characters, the customized members in SHBRJCL must be manually updated where this value is inserted into the JCL, otherwise the job might not contain valid JCL.</p> <p>Important: After you run HBRUOPTI, check the ++HBRWORKDS++.SHBRJCL data set member to ensure that the JCL is valid:</p>
++HBRWORKDS++	<p>The high-level qualifier for the working data sets that contain customized JCL for creating an execution environment instance.</p> <p>Example: HBR.WORKDS</p> <p>The value of this variable must be different from the value of the ++HBRHLQ++ variable.</p> <p>This variable is mandatory.</p>
++HBRJAVAHOME++	<p>The location of the JVM. (Java 7 is recommended, Java 6.0.1 is also supported).</p> <p>Example: /java/java7_64/J7.0_64</p>
++HBRSSPPORT++	<p>The port number on which the SSP service can be found. The value must be an integer.</p> <p>The value of this variable must be different from the value of the ++HBRCONSOLEPORT++ variable and the value of the ++HBRCONSOLECOMPOR++ variable.</p>

Table 9. Configuration variables for configuring a new Decision Server for z/OS instance. (continued)

Variable	Description
++HBRCONSOLEPORT++	<p>The port that you specify to access the console from your web browser. The value must be an integer.</p> <p>The value of this variable must be different from the value of the ++HBRCONSOLECOMPORT++ variable and the value of the ++HBRSSPPORT++ variable.</p> <p>Example: 34114</p>
++HBRCONSOLECOMPORT++	<p>The port that is used by the console to communicate with the server. The value must be an integer.</p> <p>The value of this variable must be different from the value of the ++HBRCONSOLEPORT++ variable and the value of the ++HBRSSPPORT++ variable.</p> <p>Example: 44114</p>
++HBRCONSOLECOMHOST++	<p>The host name of the machine where the console runs. Specify localhost when the console and the server that is run on the same machine.</p> <p>Example: localhost</p>
++HBRLANG++	<p>The language that is used by the server. The default value is En-US. The server uses the following valid language values:</p> <ul style="list-style-type: none"> • De_DE: German • En_US: US English • Es_ES: Spanish • Fr_FR: French • It_IT: Italian • Ja_JP: Japanese • Ko_KR: Korean • Nl_NL: Dutch • Pl_PL: Polish • Pt_BR: Brazilian Portuguese • Ru_RU: Russian • Zh_CN: Simplified Chinese • Zh_TW: Traditional Chinese <p>The supported languages are also listed in the HBRCMMN data set member.</p> <p>The value is case-sensitive and must be specified in mixed-case exactly as shown.</p>

Table 9. Configuration variables for configuring a new Decision Server for z/OS instance. (continued)

Variable	Description
++HBRTRACELEVEL++	<p>The trace level during execution. Choose from the following options:</p> <ul style="list-style-type: none"> • ALL: Logs all messages, including internal traces. • FINE: Logs debug messages, informational messages, errors, and warnings. • INFO: Logs informational messages, errors, and warnings. • WARNING: Logs errors and warnings. • SEVERE: Logs errors only. • OFF: No tracing. <p>The value is case-sensitive and must be specified in uppercase exactly as shown.</p>
++HBRPERSISTENCETYPE++	<p>The type of persistence layer that is used to store deployed artifacts. Set this variable to DB2 or FILE. The value is case-sensitive and must be specified in uppercase exactly as shown.</p>

Table 10. Configuration variables for configuring CICS to execute rules on a zRule Execution Server for z/OS instance.

Variable	Description
++CICSLIST++	<p>The CICS startup group list that is specified using the GRPLIST parameter.</p> <p>Example: DFHLIST</p> <p>The list of groups contains the resource definitions that are created when you run the ++HBRWORKDS++.<HBRSSID_NAME>.SHBRJCL(HBRCSDD) job in Configuring CICS to execute rules on zRule Execution Server for z/OS. When CICS starts, it looks at the lists of groups in the table and installs the resources that they specify.</p>

Table 11. Configuration variables for configuring zRule Execution Server for z/OS running on a CICS JVM server.

Variable	Description
++CICSHLQ++	<p>The high-level qualifier for CICS.</p> <p>Example: CTS420.CICS</p>
++CICSCSDDSN++	<p>The high-level qualifier for the CICS system.</p> <p>Example: CTS420.APPLID.DFHCSDD</p>
++JDBCPLAN++	<p>Plan that is used for JDBC connections and CICS.</p> <p>Example: DSNJCC</p>

Table 12. Configuration variables for configuring IMS to run rules on a zRule Execution Server for z/OS instance.

Variable	Description
++IMSHLQ++	The high-level qualifier for the data sets of the main IMS installation. Example: IMS.V10.DBDC
++IMSREGID++	The ID of the IMS instance to be used. Example: IM0A
++IMSREGHLQ++	The high-level qualifier of the IMS region data sets. Example: IMSDATA.IM0A

Table 13. Configuration variables for configuring a DB2 database for Operational Decision Manager for z/OS.

Variable	Description
++DB2HLQ++	The high-level qualifier for the DB2 data sets. Example: SYS2.DB2.V10
++DB2GROUP++	The RACF group that is granted access across the ODM tables. Connect ++DB2USER++ to this group.
++DB2RUNLIB++	The DB2 runtime library. Example: DSNV10GP.RUNLIB.LOAD
++DB2SUBSYSTEM++	The DB2 subsystem name. Example: DIGP
++DB2LOCATION++	The DB2 location name that is used to connect to this DB2 subsystem. Example: DSNV10GP
++DB2VCAT++	The DB2 integrated catalog facility catalog. Example: DSNV10GP
++DB2ADMIN++	The user ID authorized to create DB2 databases. Example: DB2ADMINID
++DB2SCHEMA++	The schema that is used to create and access the DB2 artifacts.
++RESDATABASE++	The name of the database that is used by the zRule Execution Server for z/OS instance. Example: RESDB1
++RTSDATABASE++	The name of the database that is used by the Decision Center instance. Example: RTSDB2

Table 13. Configuration variables for configuring a DB2 database for Operational Decision Manager for z/OS. (continued)

Variable	Description
++EVDATABASE++	The name of the database that is used by the Decision Server Events instance. Example: EVDB1
++RESSTOGROUP++	The name of the storage group that is used by the zRule Execution Server for z/OS instance. Example: RESSTG1
++EVSTOGROUP++	The name of the storage group that is used by the Decision Server Events instance. Example: EVSTG1
++RTSSTOGROUP++	The name of the storage group that is used by the Decision Center instance. Example: RTSSTG1
++DB2TABLEBP++	The buffer pool name for the tables. Example: BP1
++DB2INDEXBP++	The buffer pool name for the indexes. Example: BP2
++DB2LOBBP++	The buffer pool name for large objects. Example: BP3
++DB2SAMPLEPROGRAM++	The DB2 program name. Example: DSNTPE2
++DB2SAMPLEPROGRAMPLAN++	The DB2 plan name. Example: DSNTPE91
++DB2BP4K++	The buffer pool name for 4 K objects. Example: BP4K
++DB2BP8K++	The buffer pool name for 8 K objects. Example: BP8K
++DB2BP32K++	The buffer pool name for 32 K objects. Example: BP32K
++DB2USER++	The user ID for accessing the DB2 database.
++DB2PSWD++	The password for accessing the DB2 database.
++DB2JARLOCN++	The location of the DB2 classes. Example: /usr/lpp/db2v10/jdbc/classes

Table 14. Configuration variables for configuring WebSphere Application Server for Operational Decision Manager for z/OS.

Variable	Description
++WASINSTPATH++	The WebSphere Application Server home directory. Unique for each server instance. Example: /WebSphere/V80IL2Z1/Appserver
++WASBOOTSTRAPPORT++	Enter the bootstrap port number of the event runtime WebSphere Application Server.
++SECURITYTYPE++	Set to RACF if your WAS system has already been configured to use RACF. Set to WAS if your WAS system has already been configured to use federated security. Example: RACF
++DMGRPATH++	The Operational Decision Manager for z/OS path in a WebSphere Application Server Network Deployment environment. Example: /WebSphere/V8ILGDM The maximum field length for this variable is 59 characters. Important: After you run HBRUPTI, check the following data set members to ensure that the value is not truncated: <ul style="list-style-type: none"> • HBRDSWAS • HBRDCWAS • HBRSDVS Note: This value might be blank if a deployment manager is not being used.
++RESWASINSTANCE++	The location where the res.properties file is stored. Example: /u/HBR1/was_server1
++RTSWASINSTANCE++	The location where the rts.properties file is stored. Example: /u/HBR1/was_server1
++DVSWASINSTANCE++	The location where the dvs.properties file is stored. Example: /u/HBR1/was_server1
++WASSERVERNAME++	The WebSphere Application Server instance name. Example: Server12Base
++PROFILE++	The WebSphere Application Server profile. Set to default on z/OS. Example: default
++CELLNAME++	The WebSphere Application Server cell name. Example: cell01

Table 14. Configuration variables for configuring WebSphere Application Server for Operational Decision Manager for z/OS. (continued)

Variable	Description
++NODENAME++	The WebSphere Application Server server node name. Example: node1
++ADMINHOST++	The name of the host on which WebSphere Application Server is running.
++ADMINUSER++	The WebSphere Application Server server administration user ID.
++ADMINPSWD++	The WebSphere Application Server server administration password.
++EJBHLQ++	The SAF prefix for EJBROLES. This field might be blank. Example: CLID

Table 15. Configuration variables for enabling wsadmin scripts for Operational Decision Manager for z/OS.

Variable	Description
++RESADMIN++	The administrator user group for Rule Execution Server. Example: resAdministrators
++RESDEPLOY++	The deployer user group for Rule Execution Server. Example: resDeployers
++RESMONITOR++	The monitor user group for Rule Execution Server. Example: resMonitors
++RESADMINUSER++	An admin user for Rule Execution Server. Default value: resAdmin
++RESDEPLOYUSER++	A deployer user for Rule Execution Server. Default value: resDeployer
++RESMONITORUSER++	A monitor user for Rule Execution Server. Default value: resMonitor
++RTSADMIN++	The administrator user group for Decision Center. Default value: rtsAdministrator
++RTSCONFIG++	The configuration user group for Decision Center. Default value: rtsConfigManager
++RTSUSER++	The user group for Decision Center. Default value: rtsUser

Table 15. Configuration variables for enabling wsadmin scripts for Operational Decision Manager for z/OS. (continued)

Variable	Description
++RTSINSTALLER++	The installer user group for Decision Center. Default value: rtsInstaller
++RTSADMINUSER++	An admin user for Decision Center. Default value: rtsAdmin
++RTSCONFIGUSER++	A configuration user for Decision Center. Default value: rtsConfig
++RTSUSERUSER++	A user for Decision Center. Default value: rtsUser1
++DB2NATIVELOC++	The location of the DB2 native library files. Example: /usr/lpp/db2v10/jdbc/lib
++DB2SERVNAME++	The DB2 host name or IP address. Example: //winmvsm.ibm.com
++DB2PORT++	The DB2 connection port. Example: 40100

Table 16. Configuration variables for configuring a connection to the Rules for COBOL subsystem.

Variable	Description
++R4CSSID++	Variable to create a new subsystem ID for a COBOL rule subprogram. Example: SSID

Table 17. Configuration variables for configuring a connection to a Rule Execution Server on WebSphere Application Server through WOLA.

Variable	Description
++HBRWOLALOADLIB++	The location of the WOLA libraries to load. Example: WAS.OLA.LOADLIB
++HBRWOLACELL++	The name of the WebSphere Application Server cell to connect to using WOLA. Example: CILK
++HBRWOLANODE++	The name of the WebSphere Application Server node to connect to using WOLA. Example: NILK
++HBRWOLASERVER++	The short name of the WebSphere Application Server instance to connect to using WOLA. Example: WSVRILK

Related tasks:

“Step 5: Creating the working data sets” on page 14

After you have customized the variables for your site, you must then create the working data sets for your topology.

Related information:

“Checking the existing setup” on page 34

You must confirm that WebSphere Application Server for z/OS uses the RACF local operating system security repository before proceeding. Security is configured when the application server is installed.

“Step 2: Choosing your topology” on page 2

You can configure Operational Decision Manager for z/OS in different topologies that are based on your production needs and your existing configurations.

“Step 5: Creating the working data sets” on page 14

After you have customized the variables for your site, you must then create the working data sets for your topology.

z/OS runtime variables

The runtime variables that you use for setting up the Operational Decision Manager for z/OS runtime are stored in different data set members.

Configuration tasks and data set members

Table 18. Configuration tasks and data set members

Task	Data set member	Variables
Configure batch or IMS jobs on zRule Execution Server for z/OS.	“HBRBATCH” on page 128	Variables that enable batch applications to connect to the correct zRule Execution Server for z/OS. The HBRBATCH data set member belongs to SHBRPARM.
Configure and access a DB2 database persistence layer for zRule Execution Server for z/OS running on a CICS JVM server.	“HBRCICSD” on page 128	Variables for a DB2 database that is used as the persistence layer for zRule Execution Server for z/OS running on a CICS JVM server. The HBRCICSD data set member belongs to SHBRPARM.
Configure CICS applications to run on zRule Execution Server for z/OS running on a CICS JVM server.	“HBRCICSJ” on page 129	Variables that are used for CICS applications on a zRule Execution Server for z/OS running on a CICS JVM server. The HBRCICSJ data set member belongs to SHBRPARM.
Configure CICS applications to run on zRule Execution Server for z/OS.	“HBRCICSZ” on page 130	Variables that are used for CICS applications on zRule Execution Server for z/OS. The HBRCICSZ member belongs to SHBRPARM.
Configure communications with the console, and customization of trace level.	“HBRMCMN” on page 130	Variables that are used by the server to communicate with the console, and to customize the trace level. The HBRMCMN data set member belongs to SHBRPARM.

Table 18. Configuration tasks and data set members (continued)

Task	Data set member	Variables
Configure the event run time.	"HBREV" on page 131	Variables that are used for applying event properties to a WebSphere Application Server instance. These values must be reviewed before submission with the HBREVAUG JCL job. The HBREV data set member belongs to SHBRWASC.
Configure server startup.	"HBRMSTR" on page 133	Variables that are used to start a server instance. The HBRMSTR data set member belongs to SHBRPARM.
Configure and access the persistence layer.	"HBRPSIST" on page 134	Variables that specify persistence type information. The HBRPSIST data set member belongs to SHBRPARM.
Configure batch, IMS,, or CICS applications to run on Rule Execution Server on WebSphere Application Server through WOLA.	"HBRWOLA" on page 135	Variables that enable client applications to connect to a Rule Execution Server on WebSphere Application Server, through a WebSphere Optimized Local Adapter (WOLA). The HBRWOLA data set member belongs to SHBRPARM.
Not customizable.	HBRSCEN	Variables that specify the borrower input values for the Miniloan sample application. This data set member requires no customization. The HBRSCEN data set member belongs to SHBRPARM.

HBRBATCH

Table 19. HBRBATCH variables for configuring batch or IMS jobs on zRule Execution Server for z/OS.

Variable	Description
HBRSSIDLIST	<p>A zRule Execution Server for z/OS server that consists of a list of 1 to 32 subsystem IDs separated by commas. For example: HBR1,HBR2,HBR3. The first ID in the list is the primary server to which rule execution is routed. Other servers run rule sets only if rule execution is transferred to them. To route rule execution to a particular server, specify its ID first.</p> <p>Note: Instances of zRule Execution Server for z/OS running on a CICS JVM server can be configured only as a server group with a single server.</p> <p>Tip: If you give each server the same subsystem ID prefix, you can use wildcards to apply commands to multiple servers. For example, if each server name begins with HBR, you can use the wildcard HBR% to specify multiple servers on RACF commands rather than issue the same command for each server. For example, RDEFINE HBRCONN HBR% UACC(NONE)</p>
HBRTARGETRES	<p>This variable specifies the location for the rule execution.</p> <p>In the example, ZRES means that the rules run in a standard zRule Execution Server for z/OS.</p> <p>For example: HBRTARGETRES=ZRES</p>

HBRICSD

Table 20. HBRICSD variables for configuring and accessing a DB2 database persistence layer for zRule Execution Server for z/OS running on a CICS JVM server.

Variable	Description
HBRPERSISTENCETYPE	The type of persistence layer that is used to store deployed artifacts. This variable is pre-set to DB2. Do not change the value.
HBRDBURL	The execution environment that is running in CICS uses this URL to access the DB2 connection that is established by CICS.

HBRCICSJ

Table 21. HBRCICSJ variables for configuring CICS applications to run on zRule Execution Server for z/OS running on a CICS JVM server.

Variable	Description
<i>HBRWORKPATH</i>	<p>The location of the working directory in the UNIX System Services file system.</p> <p>Example: /u/HBR1</p> <p>Initialized by the ++<i>HBRWORKPATH</i>++ variable that is specified in the control statement.</p>
<i>HBRINSTPATH</i>	<p>The location of the installation root directory in the UNIX System Services file system.</p> <p>Example: /usr/lpp/zDS/V8R6M0</p> <p>Initialized by the ++<i>HBRINSTPATH</i>++ variable in the HBRINST member.</p>
<i>HBRTARGETRES</i>	<p>This variable specifies the location for the rule execution.</p> <p>The only possible value for running in rules in a CICS that hosts a CICS JVM server is LCICSJVM. This value is used because all rules run locally in the CICS JVM server.</p> <p>For example: HBRTARGETRES=LCICSJVM</p>
<i>HBRPOOLMAXSIZE</i>	<p>This variable controls the size of the <code>pool.maxSize</code> setting for the <code>defaultConnectionManagerProperties</code> setting in <code>ra.xml</code>. It controls the number of concurrent threads that can run rule sets.</p> <p>If it is defined, the value of <i>HBRPOOLMAXSIZE</i> must be higher than the <code>ThreadLimit</code> setting on the HBRJVM JVM server that is defined in CICS. The default value is 20 if this variable is not set.</p>

HBRCICSZ

Table 22. HBRCICSZ variables for configuring CICS applications to run on zRule Execution Server for z/OS.

Variable	Description
HBRSIDLIST	<p>A zRule Execution Server for z/OS server group that consists of a list of 1 to 32 subsystem IDs separated by commas. For example: HBR1,HBR2,HBR3. The first ID in the list is the primary server, from which you start the shared console. Rule execution is routed to the first available server in the list. Other servers run rule sets only if rule execution is transferred to them. To route rule execution to a particular server, specify its ID first.</p> <p>Note: Instances of zRule Execution Server for z/OS running on a CICS JVM server can be configured only as a server group with a single server.</p> <p>Tip: If you give each server the same subsystem ID prefix, you can use wildcards to apply commands to multiple servers. For example, if each server name begins with HBR, you can use the wildcard HBR% to specify multiple servers on RACF commands rather than issue the same command for each server. The RDEFINE HBRCONN HBR% UACC(NONE) command is an example that uses a wildcard.</p>
HBRTARGETRES	<p>This variable specifies the location for the rule execution.</p> <p>There are two possible values for running rules in a CICS region:</p> <p style="padding-left: 40px;">ZRES RCICSJVM</p> <p>The rule execution can be used on another CICS region that acts as an owning region for that rule, or used on a zRule Execution Server for z/OS on the same LPAR.</p> <p>For example: HBRTARGETRES=RCICSJVM</p>

HBRCMMN

Table 23. HBRCMMN runtime variables for configuring communicating with the console, and for customizing trace level.

Variable	Description
HBRLANG	<p>The language that is used by the server. The list of supported languages is in the HBRCMMN data set member.</p>

Table 23. HBRCCMN runtime variables for configuring communicating with the console, and for customizing trace level. (continued)

Variable	Description
<i>HBRCCSID</i>	Variable that is related to code page conversion. Example: 1047
<i>HBRCONSOLECOM</i>	This variable indicates whether the zRule Execution Server for z/OS is to connect to the Rule Execution Server console for management purposes. <ul style="list-style-type: none"> • Set to YES to enable a management connection to the Rule Execution Server console. • Set to NO for situations where a console connection is not required, for example, Java Batch execution.
<i>HBRCONSOLECOMPORT</i>	The port that is used by the console to communicate with the server. Example: 44114
<i>HBRCONSOLECOMHOST</i>	The host name of the LPAR where the console is running. Specify localhost when the console and the server are running on the same LPAR. Example: localhost
<i>HBRTRACELEVEL</i>	The trace level during execution. Choose from the following options: <ul style="list-style-type: none"> • ALL: Logs all messages, including internal traces. • FINE: Logs debug messages, informational messages, errors, and warnings. • INFO: Logs informational messages, errors, and warnings. • WARNING: Logs errors and warnings. • SEVERE: Logs errors only. • OFF: No tracing.

HBREV

Table 24. HBREV variables for configuring event run time.

Variable	Description
<i>profileName</i>	The name of the existing application server profile that you plan to augment. The default value is default.
<i>templatePath</i>	The location of the profile template. Set the value to ++HBRINSTPATH++/events/config/wbeProfileTemplate/wbe/default.
<i>profilePath</i>	The location of the profile. Set the value to ++WASINSTPATH++/profiles/default.

Table 24. HBREV variables for configuring event run time. (continued)

Variable	Description
<i>adminUserName</i>	The user ID for accessing WebSphere Application Server. Set the value to ++ADMINUSER++. This variable is required only if administrative security is enabled.
<i>adminPassword</i>	The password for accessing WebSphere Application Server. Set the value to ++ADMINPSWD++ This variable is required only if administrative security is enabled.
<i>hostName</i>	The host name of the LPAR that hosts the profile. Set the value to ++ADMINHOST++.
<i>wbeHome</i>	The fully qualified path of the directory where Decision Server Events is installed. Set the value to ++HBRINSTPATH++/events.
<i>wbeMsgingType</i>	The JMS provider to be configured for WebSphere Application Server. Choose from the following options: <ul style="list-style-type: none"> Default_Messaging: The default messaging provider. This type of messaging is the default. MQ_JMS_Messaging: WebSphere MQ is the messaging provider. No_Messaging: No JMS provider is to be configured. The JMS provider is configured manually.
<i>wbeDbType</i>	The DB2 database that you use. Set the value to DB2_Universal.
<i>wbeDbJDBCClasspath</i>	The path to the directory that contains db2jcc.jar and db2jcc_license. Set to ++DB2JARLOCN++/db2jcc.jar.
<i>wbeDbHostName</i>	The host name for the database server. Set the value to ++DB2SERVNAME++.
<i>wbeDbServerPort</i>	The port where the TCP/IP service is assigned or the port on which the database is listening. Set the value to ++DB2PORT++.
<i>wbeDbName</i>	The name of the database. Set the value to ++DB2VCAT++
<i>wbeDbUserId</i>	The user ID for accessing the DB2 database. Set the value to ++DB2USER++.
<i>wbeDbPassword</i>	The password for accessing the DB2 database. Set the value to ++DB2PSWD++.
<i>wbeDbCurrSQLID</i>	The current SQL ID. Set the value to ++DB2CURRSQLID++.

HBRMSTR

Table 25. HBRMSTR variables for configuring server startup.

Variable	Description
<i>HBRMODE</i>	<p>Choose from the following options:</p> <ul style="list-style-type: none"> • RULE: The server accepts and processes rule executions from local clients. This mode is used when you run and manage rules from a mainframe. • CONSOLE: The server starts the RES Console for managing the other servers in the group. The server does not accept connections from local clients. • TEST: The server is being used for testing and does not accept connections from local clients. This mode is used to start SSP and run rules only from the console. <p>The value is case-sensitive and must be specified in uppercase exactly as shown.</p>
<i>HBRSSID</i>	<p>The ID of the subsystem where the zRule Execution Server for z/OS instance runs.</p> <p>Example: HBR1</p>
<i>HBRWORKPATH</i>	<p>The location of the working directory in the UNIX System Services file system.</p> <p>Example: /u/HBR1</p> <p>Initialized by the ++<i>HBRWORKPATH</i>++ variable in the HBRINST member.</p>
<i>HBRINSTPATH</i>	<p>The location of the installation root directory in the UNIX System Services file system.</p> <p>Example: /usr/lpp/zDS/V8R6M0</p> <p>Initialized by the ++<i>HBRINSTPATH</i>++ variable in the HBRINST member.</p>
<i>HBRCONSOLEPORT</i>	<p>The port number on which the console can be found.</p> <p>Initialized by ++ and used when <i>HBRMODE</i>=CONSOLE.</p>
<i>HBRSCOPE</i>	<p>To prevent multiple subsystems from running with the same SSID, an ENQ is obtained at subsystem startup. By default, the ENQ obtained has sysplex-wide scope so you cannot have two subsystems on different LPARs with the same SSID. The <i>HBRSCOPE</i> variable can be used to change this behavior so that only the ENQ has LPAR scope. Ensure that only the ENQ has LPAR scope by setting <i>HBRSCOPE</i>=LPAR. Set <i>HBRSCOPE</i>=PLEX if you want to make the default explicit.</p>

Table 25. HBRMSTR variables for configuring server startup. (continued)

Variable	Description
HBRSSPPORT	The port number on which the SSP service can be found. Initialized by ++ and used when HBRMODE=TEST.
HBRARMENABLED	Indicates whether the server is to be managed by Automatic Restart Manager (ARM). <ul style="list-style-type: none"> Set to YES to have service disruptions that are managed by ARM. Set to NO otherwise.
JAVA_OPTIONS	The arguments to pass to the JVM when the zRule Execution Server for z/OS starts up. Details are provided as comments in the data set member. Example: -Xmx768M -Xms768M
JAVA_HOME	The location of the JVM. Example: /java/J7.0_64 Initialized by the ++HBRJAVAHOME++ variable in the HBRINST member.
LIBPATH_SUFFIX	Extra libpath entries.
CLASSPATH_SUFFIX	Extra classpath entries.
HBRSMFST100	Indicates whether zRule Execution Server for z/OS is to collect execution data in the System Management Facility (SMF) records. <ul style="list-style-type: none"> Set to YES to enable the SMF recording feature. Set to NO otherwise.

HBRPSIST

The HBRPSIST data set member includes the following variables:

- Table 26
- Table 27 on page 135
- Table 28 on page 135

Table 26. HBRPSIST variables for configuring and accessing the persistence layer.

Variable	Description
HBRPERSISTENCETYPE	The type of persistence layer that is used to store deployed artifacts. Set this variable to DB2 or FILE.

Table 27. HBRPSIST variables for configuring a file system as the persistence layer.

Variable	Description
HBRFILEPATH	The location where the ruleApps are stored when a file system is used as the persistence layer. Example: /u/HBR1/res_xom Initialized by the ++HBRWORKPATH++ variable in the HBRINST member.
HBRXOMFILEPATH	The location where the XOMs are stored when a file system is used as the persistence layer. Example: /u/HBR1/res_xom Initialized by the ++HBRWORKPATH++ variable in the HBRINST member.

Table 28. HBRPSIST variable for configuring a DB2 database as the persistence layer.

Variable	Description
HBRDBURL	The URL for accessing the DB2 database.

HBRWOLA

Table 29. HBRWOLA variables for configuring batch jobs to run on Rule Execution Server on WebSphere Application Server through WOLA.

Variable	Description
HBRWOLACELL	The name of the WebSphere Application Server cell to connect to using WOLA. Example: CILK
HBRWOLANODE	The name of the WebSphere Application Server node to connect to using WOLA. Example: NILK
HBRWOLASERVER	The short name of the WebSphere Application Server server to connect to using WOLA. Example: WSVRILK
HBRWOLACICS2WAS	To use the CICS application task identity in the WebSphere Application Server authorization process, add <i>HBRWOLACICS2WAS= YES</i> to HBRWOLA. Adding this variable sets the WOLA registration flag, reg_flag_C2Wprop , to true. For more information, see Optimized local adapters environment variables in the WebSphere Application Server Information Center. Example: HBRWOLACICS2WAS = YES

Table 29. HBRWOLA variables for configuring batch jobs to run on Rule Execution Server on WebSphere Application Server through WOLA. (continued)

Variable	Description
<i>HBRTARGETRES</i>	<p>This variable specifies the location for the rule execution.</p> <p>When you run rules in a WebSphere Application Server by using a WOLA configuration, the only possible value for this variable is WOLA.</p> <p>Example: HBRTARGETRES=WOLA</p>
<i>HBRWOLAREGNAME</i>	<p>This variable can be used to set the WOLA registration name as described in the documentation for BBOA1REG. The default, when this variable is not set, is to call BBOA1REG when a connection is first made by using a randomly created name. The alternative is to do the registration yourself (for example under CICS by calling BBOC REG), then pass the registration name to Operational Decision Manager for z/OS using this variable (in which case no calls to BBOA1REG/BBOA1URG are carried out on connect/disconnect).</p> <p>Make the value a maximum of 12 characters long.</p> <p>Example: HBRWOLAREGNAME=ILKREGNAME</p>

z/OS data sets and directories

The SMP/E installation creates a number of data sets and directories on your z/OS system.

The following table lists the data sets that are installed. ++*HBRHLQ*++ represents the high-level qualifier assigned to the data sets during installation.

Data set	Description
++ <i>HBRHLQ</i> ++.SHBRAUTH	APF-authorized load modules.
++ <i>HBRHLQ</i> ++.SHBRCICS	CICS-specific load modules.
++ <i>HBRHLQ</i> ++.SHBRCOBC	COBOL copybooks.
++ <i>HBRHLQ</i> ++.SHBRCOBS	COBOL sample code.
++ <i>HBRHLQ</i> ++.SHBREXEC	Installation-specific executable files.
++ <i>HBRHLQ</i> ++.SHBRINST	SMP/E jobs.
++ <i>HBRHLQ</i> ++.SHBRJCL	Runtime samples and configuration jobs.
++ <i>HBRHLQ</i> ++.SHBRLOAD	Product load modules.
++ <i>HBRHLQ</i> ++.SHBRPARAM	Runtime configuration parameters.
++ <i>HBRHLQ</i> ++.SHBRPLIC	PL/I include files.

Data set	Description
++HBRHLQ++.SHBRPLIS	PL/I sample code.
++HBRHLQ++.SHBRPROC	Runtime JCL procedures.
++HBRHLQ++.SHBRWASC	Property files for WebSphere Application Server for z/OS installation.
++HBRHLQ++.SHBRXLCH	C sample header files.
++HBRHLQ++.SHBRXLCS	C source code.

The following table lists the directories that are installed for Decision Server for z/OS:

Directory	Description
/usr/lpp/zDM/V8R6M0/executionserver	Runtime files for Rule Execution Server on WebSphere Application Server for z/OS.
/usr/lpp/zDM/V8R6M0/zexecutionserver	Runtime files for zRule Execution Server for z/OS.
/usr/lpp/zDM/V8R6M0/events	Runtime files for the event runtime.

The following table lists the UNIX System Services (USS) directories that are installed for Decision Center for z/OS:

Directory	Description
/usr/lpp/zDM/V8R6M0/teamserver	Files for Decision Center on WebSphere Application Server for z/OS.

You must set the variables for your site before you can run any of the runtimes or start to use the Decision Center for z/OS consoles.

HBRUOPTI job

The ++HBRHLQ++.SHBRJCL(HBRUOPTI) job creates working data sets based on the control statements that are specified in the ++HBRHLQ++.SHBRPARG(HBRCTRL) data set member.

The JCL variables you set in the ++HBRHLQ++.SHBRPARG(HBRINST) data set member are taken and combined with the variables provided in the control statements to produce the working data sets for configuring a topology.

The HBRUOPTI job performs the following tasks:

- Copies the required data sets and members and assigns them the high-level qualifier represented by the ++HBRWORKDS++ JCL variable.
- Assigns the JCL variables in ++HBRHLQ++.SHBRPARG(HBRINST) or ++HBRHLQ++.SHBRPARG(HBRCTRL) to all occurrences of the corresponding runtime variable in the data set members.

For example, ++WASINSTPATH++ represents the installation path for the WebSphere Application Server instance. If you set ++WASINSTPATH++ to the value /WebSphere/V80IL2Z1/AppServer in HBRINST, the corresponding runtime variable, WASINSTPATH, is set to the same value wherever it is referenced.

++HBRSSID++ represents the SSID of the zRule Execution Server for z/OS. If you set HBRSSID=HBR1 in HBRCTRL, the corresponding runtime variable, *HBRSSID*, is set to HBR1 wherever it is referenced.

Related information:

“Step 5: Creating the working data sets” on page 14

After you have customized the variables for your site, you must then create the working data sets for your topology.

“Step 4: Customizing the control statements for your topology” on page 9

Control statements tell the HBRUUPTI job which execution environments to create. When you customize the z/OS configuration and runtime variables, you must customize the control statements for your topology. You specify control statements in the ++HBRHLQ++.SHBRP(HBRCTRL) data set member.

Notices

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

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

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

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England SO21 2JN

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England SO21 2JN

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

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

imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

Index

A

- access rights
 - for MBean descriptors 48
- asynchronous execution
 - support in Decision Server 59

B

- batch 8

C

- cells
 - Rule Execution Server configuration in a cluster environment 66
- changing, class loading sequence
 - Decision Center 90
- CICS
 - configuring forzRule Execution Server for z/OS 19, 29
 - executing rules on Rule Execution Server on WebSphere Application Server for z/OS through WOLA 56
 - wrapper program, calling 103
- CICS application-owning region
 - application-owning region 32
 - application-owning region 32
 - configuring to execute rules in a rule-owning region 32
 - rule-owning region 32
- class loading sequence, changing
 - Decision Center 90
- clusters
 - WebSphere Application Server for z/OS 70
- COBOL dynamic link 102
- COBOL static link 101
- code integration
 - CICS wrapper program 103
 - using a dynamic link 102
 - using a static link 101
- command security
 - zRule Execution Server for z/OS 27
- compiling generated code 109
- configuration
 - for z/OS 106
 - Resource Manager JCL 108
- configure HBRCTRL 9, 10, 11, 12, 13
- configure Java execution units to connect to the zRules Execution Server console 53
- configuring
 - creating a connection pool 87
 - creating a data source 87
 - federated repository, using 37
 - zRule Execution Server for z/OS 18
 - zRule Execution Server for z/OS on CICS JVM server 30
- configuring Decision Center
 - class loading sequence, changing 90

- configuring Decision Center (*continued*)
 - creating a data source 86
 - creating a JDBC provider 86
 - security policies 85
- configuring Decision Validation Services
 - redeploying Decision Validation Services 93
- configuring Rule Execution Server 46
 - creating a data source 46
 - deploying the hosted transparent decision service EAR 51
 - deploying the management EAR 50
 - deploying the XU RAR 48
 - environments 66
 - resgroup to the Monitor role, mapping 40
 - using WebSphere Application Server for z/OS in cluster mode 70
 - WebSphere Application Server for z/OS
 - WebSphere Optimized Local Adapters (WOLA) 53
- connection factories
 - for WebSphere MQ input and output queues 60
 - for WebSphere MQ input and output topics 62
- connection pool
 - creating 87
- connection security
 - zRule Execution Server for z/OS 25
- console security
 - zRule Execution Server for z/OS 26
- control statement 9, 10, 11, 12, 13
- creating
 - Rule Execution Server 46
- custom properties
 - Decision Center security 82, 85
- customizing JCL variables 9

D

- data sets 136
- data source
 - creating 87
- data source, creating
 - Rule Execution Server 46
- database
 - testing connection 46, 89
- DB2 persistence layer
 - zRule Execution Server for z/OS 15
- Decision Center 89
 - opening 97
 - security
 - custom properties 82, 85
- Decision Server for z/OS
 - load library 1
 - securing directories 22
 - working data sets 14

- Decision Validation Services
 - configuring
 - on WebSphere Application Server for z/OS 91
 - deploying on z/OS 91
 - redeploying 93
 - deploying 51
 - WebSphere Application Server for z/OS management EAR 50
 - deploying on z/OS
 - Decision Validation Services 91
 - deploying the EAR file 89
 - deploying the XU RAR 48
 - deploying, hosted transparent decision service EAR
 - Rule Execution Server 51
 - deploying, management EAR
 - Rule Execution Server 50
 - deploying, XU RAR,
 - Rule Execution Server 48
- diagnostics
 - Rule Execution Server console 99
- directories 136
- disabling security
 - zzRule Execution Server for z/OS 24

E

- EAR file, deploying 89
- environments
 - Rule Execution Server configuration 66
- executing rules in batch
 - using WebSphere Optimized Local Adapters (WOLA) 56
- execution data 111

F

- federated repository, using 37

G

- generated COBOL code
 - compiling 109
 - executing 109
 - linking 109
- Get Started for Sub-capacity Pricing 105
- GSSP 105

H

- HBRBATCH 117, 126
- HBRCICSD 117, 126
- HBRCICSJ 117, 126
- HBRCICSZ 117, 126
- HBRCMMN 117, 126
- HBRCTRL 9, 10, 11, 12, 13, 117
- HBREV 117, 126

HBRINST 9, 117, 126
HBRMSTR 117, 126
HBRPSIST 117, 126
HBRUUPI 14, 137
HBRUUPI job 9, 13, 137
HBRWOLA 117, 126
hosted transparent decision service EAR,
deploying
Rule Execution Server 51

I

IMS
batch processing region 20, 58
configuration 20, 58
DL/I 20, 58
message processing region 20, 58
supported versions 20, 58
input/output queues
creating the connection factory in
WebSphere MQ 60
input/output queues, WebSphere MQ
activation specification 63
input/output topics, WebSphere MQ
activation specification 64
creating the connection factory 62
installing
database, testing connection 89
installing Rule Execution Server
creating a JDBC provider 45

J

J2C authentication alias
WebSphere Application Server for
z/OS
Rule Execution Server 45
JCL
for Resource Manager
configuration 108
JCL variables 9, 117, 126
JDBC provider, creating
Decision Center 86
Rule Execution Server 45

L

Last Participant Support (LPS)
enabling in WebSphere MQ 66
linking generated code 109
load library
zRule Execution Server for z/OS 1

M

management EAR, deploying
Rule Execution Server 50
mapping, resgroup to the Monitor role
Rule Execution Server 40
MBeans
and WebSphere MQ 59
descriptors for WebSphere Application
Server 7.0 48

message-driven rule beans
and WebSphere MQ
installation 65
queue activation specification 63
topic activation specification 64

MQ

Integrating WebSphere MQ in
WebSphere Application Server 59

N

naming constraints
data sources
WebSphere Application Server for
z/OS 70

O

opening
Decision Center 97

P

permissions
for MBean descriptors 48
publishing rule projects
from Decision Center 96

Q

queues, WebSphere MQ
for request messages 61
for response messages 61

R

RACF classes for server security
zRule Execution Server for z/OS 23
redeploying
Decision Validation Services 93
resgroup to the Monitor role, mapping
Rule Execution Server 40
Resource Manager 105, 107
rule execution 100
Rule Execution Server on WebSphere
Application Server for z/OS
configuring IMS 58
rule projects
publishing
from Decision Center 96
rule sessions
and WebSphere MQ integration 59
running generated code 109
runtime variables 117, 126

S

securing directories
Decision Server for z/OS 22
security
configuring for WebSphere
Application Server for z/OS 44
configuring for WebSphere Optimized
Local Adapters (WOLA) 44

security (*continued*)
custom properties 82, 85
Decision Center 82, 85
security options
zRule Execution Server for z/OS 22
security policies
Decision Center 85
server groups
configuring 18
server security
zRule Execution Server for z/OS 23
SHBRPARAM 117, 126
SHBRWASC 117, 126
SMF 105, 111
SMF data collection
sample code 110
SMF type 120 111
SMF type 120 subtype 100 111
System Management Facility 105, 111

T

topics, WebSphere MQ
for request messages 62
for response messages 63
topologies
Operational Decision Manager for
z/OS 2, 18
zRule Execution Server for z/OS 2,
4, 7
transaction management
in WebSphere Application Server, Last
Participant Support 66

U

usage data collection
sample code 110

V

variables
WebSphere Application Server for
z/OS 45

W

WebSphere Application Server
Rule Execution Server cell
deployment 69
WebSphere Application Server for z/OS
clusters 70
configuring WOLA load libraries 53
security, configuring 44
WebSphere Application Server for z/OS
variables 45
WebSphere MQ
input queue, creating 61
input topic, creating 62
integrating in WebSphere Application
Server 59
Last Participant Support, enabling 66
message-driven rule bean,
installing 65
output queue, creating 61

- WebSphere MQ (*continued*)
 - output topic, creating 63
 - queue activation specification, creating 63
 - queue connection factory, creating 60
 - topic activation specification, creating 64
 - topic connection factory, creating 62
- WebSphere Optimized Local Adapters (WOLA) 53
 - deploying the EJB 54
 - executing rules in batch 56
 - executing rules in CICS 56
 - security, configuring 44
 - verifying the configuration 59
- WOLA
 - See* WebSphere Optimized Local Adapters (WOLA)
- WOLA load libraries
 - configuring on WebSphere Application Server for z/OS 53
- working data sets
 - Decision Server for z/OS 14
- wsadmin scripts
 - configuring Decision Center console on WebSphere Application Server for z/OS 81
 - configuring Rule Execution Server on WebSphere Application Server for z/OS 41
- zRule Execution Server for z/OS on CICS
 - JVM server supported versions
 - CICS 30
- zRule Execution Server for z/OS security
 - server 23
- zzRule Execution Server for z/OS on CICS JVM server
 - configuring 30
- zzRule Execution Server for z/OS security
 - disabling 24

X

- XOM data source 46
- xomdatasource 46
- XU RAR, deploying
 - Rule Execution Server 48
- XUConnectionFactory
 - default JNDI name 66

Z

- z/OS
 - WebSphere Optimized Local Adapters (WOLA) 53
- z/OS configuration variables 9
- z/OS subsystem 107
- zRule Execution Server for z/OS
 - command security 27
 - configuring 18
 - configuring CICS 19
 - configuring CICS ROR and AOR 29
 - configuring IMS 20
 - connection security 25
 - console security 26
 - DB2 persistence layer 15
 - diagnostics 99
 - embedded 8
 - RACF classes for server security 23
 - security options 22
 - starting 29
 - topologies 2, 4, 7, 8
 - verifying configuration 99
- zRule Execution Server for z/OS for z/OS
 - topologies 2, 18



Printed in USA