IBM Cloud 9 for SCLM for z/OS

Installation Guide

Version 2 Release 1

SC31-8845-05

IBM Cloud 9 for SCLM for z/OS

Installation Guide

Version 2 Release 1

SC31-8845-05

Note
Before using this document, read the general information under "Notices" on page 223.

Sixth edition (April 2004)

This edition applies to Version 2 Release 1 of the licensed program IBM Cloud 9 for SCLM for z/OS (program number 5655-G93) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications by telephone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The telephone number is (800) 879-2755. The fax number is (800) 284-4721.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the following address.

If you would like to make comments about this publication, address them to:

```
IBM Corporation
Department J46A/G4
555 Bailey Ave
San Jose, CA 95141-1099 U.S.A.

or fax your comments from within the U.S.A.,
to 800-426-7773, or, from ouside the U.S.A., to 408-463-2629.
```

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

```
Title and order number of this book
Page number or topic related to your comment
```

Contents

Figures

Tables

About this document

This document contains the installation procedure for the IBM Cloud 9 for SCLM for z/OS product, which combines standard z/OS installation procedures with UNIX System Services (USS) and IBM HTTP server configuration.

There are also sections on enabling the Cloud 9 S-FTP and S-JDK Java Development Kit components.

Who should use this document

This document is written for system programmers who are configuring and administering the Cloud 9 Web server. Readers should be familiar with the UNIX System Services (USS) environment, Hierarchical File System (HFS) structure, Resource Access Control Facility (RACF) profiles needed to support USS and started tasks (or the equivalent for the installed security product), and the IBM HTTP Server.

It also contains information to be used by the administrator of any SCLM projects that are using the Java and USS component languages. These administrators also need to be familiar with the USS environment and HFS structures, REXX Script, and the Java Compiler and SCLM project and language definitions.

Where to find more information

Where necessary, this document references information in other books, using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of z/OS, see *z/OS Information Roadmap* (GC28-1727). Direct your request for copies of any IBM publication to your IBM representative or to the IBM branch office serving your locality.

There is also a toll-free customer support number (1-800-879-2755) available Monday through Friday from 6:30 a.m. through 5:00 p.m. Mountain Time. You can use this number to:

- Order or inquire about IBM publications
- Resolve any software manufacturing or delivery concerns
- Activate the program reorder form to provide faster and more convenient ordering of software updates

Hardcopy publications

Table 1. Hardcopy Publications

Short Title Used in This Document	Title of Publication	Order Number
HTTP Server Guide	IBM HTTP Server for OS/390 HTTP Server Planning, Installing, and Using	SC31-8690-xx
USS Planning	OS/390 UNIX System Services Planning	SC28-1890-xx
USS Messages	OS/390 UNIX System Services Messages and Codes	SC28-1908-xx

Table 1. Hardcopy Publications (continued)

Short Title Used in This Document	Title of Publication	Order Number
USS Commands	OS/390 UNIX System Services Command Reference	SC28-1892-xx
SCLM Project Manager's Guide	Interactive System Productivity Facility (ISPF) Software Configuration and Library Manager (SCLM) Developer's and Project Manager's Guide	SC34-4750-xx
SCLM Reference	Interactive System Productivity Facility (ISPF) Software Configuration and Library Manager (SCLM) Reference	SC28-1320-xx
Breeze Installation Guide	IBM Breeze for SCLM for z/OS Installation Guide	SC31-8819-01

Softcopy publications

The z/OS library is available on the z/OS Collection Kit, SK2T-6700. This softcopy collection contains a set of z/OS and related unlicensed product books. The CD-ROM collection includes the IBM Library Reader, a program that customers can use to read the softcopy books.

Softcopy z/OS publications are also available for Web browsing. PDF versions of the z/OS publications for viewing or printing using Adobe Acrobat Reader are available at these URLs:

```
http://www.ibm.com/s390/os390/
http://www.ibm.com/servers/eserver/zseries/zos
```

Select "Library."

IBM Systems Center publications

IBM Systems Center produced Redbooks that can be helpful in setting up and using z/OS UNIX System Services. You can order these publications through the usual channels, or you can view them with a Web browser from this URL:

```
http://www.redbooks.ibm.com
```

These books have not been subjected to any formal review nor have they been checked for technical accuracy, but they represent current product understanding (at the time of their publication) and provide valuable information about a wide range of z/OS topics. You must order them separately. A selected list of these books follows:

Table 2. IBM Systems Center Publications

Title of Publication	Order Number	Comments
P/390, R/390, S/390 Integrated Server: OS/390 New User's Cookbook	SG24-4757-01	Despite the title, it is oriented toward the system programmer, and describes considerations for the UNIX System Services environment

Table 2. IBM Systems Center Publications (continued)

Title of Publication	Order Number	Comments
Debugging UNIX System Services, Lotus Domino, Novell Network Services, and other Applications on OS/390	SG24-5613-00	Provides an overview of the UNIX System Services environment along with tips and suggestions for setup and problem analysis.
OS/390 e-business Infrastructure: IBM HTTP Server 5.1 - Customization and Usage	SG24-5603-00	Provides an overview of Web servers in general with specific details for the OS/390 server along with hints and tips for setup and customization
Debugging UNIX System Services	SG24-5613-00	
e-business Enablement Cookbook for OS/390 Volumes 1,2, and 3	SG24-5664-00 SG24-5981-00 SG24-5980-00	

Summary of changes

This chapter describes the changes made in the documentation supporting Cloud 9 Version 2.1. Technical changes to the document are marked in the text by a vertical change bar in the left margin.

Important Information
The current edition of this document describes Cloud 9 V2.1 with the PTFs for APAR OA03122 applied. These PTFs must be applied before using Cloud 9 V2.1 or the product will not behave as described in this document.

Sixth Edition (April 2004)

The following enhancements and changes have been made to the installation process for Cloud 9 for SCLM for z/OS Version 2 since the last release of the product.

- Changes to translators and control files used in the customization of the S-JDK for USS Build and Deploy and FTP Remote Build and Deploy.
- The Cloud 9 WebSphere Studio Application Developer Plug-In (WSAD Plug-in) has been added to the product, covered in a new section in the Installation Guide.

Minor changes have been made to this Edition of the Installation Guide, to reflect the latest JCL samples and provide improved explanations.

Fifth Edition (April 2003)

The following enhancements and changes have been made to the installation process for Cloud 9 for SCLM for z/OS Version 2 since the last release of the product.

- Changes to translators and control files used in the customization of the S-JDK for USS.
- S-JDK for FTP Remote Build and Deploy feature has been added to the product, covered in a new section in the Installation Guide.
- The Cloud 9 Visual Age for Java Plug-in has been added to the product, covered in a new section in the Installation Guide.
- Restructuring and standardization of installation steps and Installation Guide documentation.

Minor changes have been made to this Edition of the Installation Guide, to reflect the latest JCL samples and provide improved explanations.

Installation overview

This manual contains the installation procedure for all components of the IBM Cloud 9 for SCLM for z/OS product. The procedure is a combination of standard z/OS installation procedures, UNIX System Services HFS file set up, and IBM HTTP server configuration. Hereinafter, the following names are used in this manual:

- IBM Cloud 9 for SCLM for z/OS is called ***Cloud 9***.
- IBM Breeze for SCLM for z/OS is called ***Breeze***.
- The IBM z/OS HTTP server is called ***the HTTP server***.
- UNIX System Services and HFS are called ***USS***.
- The SCLM-Java Development Kit is called S-JDK.

The manual is structured into 6 main parts:

- Part 1: Installing the Cloud 9 base product
- Part 2: Installing the S-JDK for UNIX System Services
- Part 3: Installing the S-JDK for FTP remote build and deploy
- Part 4: Installing the S-FTP feature for remote build and deploy
- Part 5: Installing the Cloud 9 VisualAge for Java Plug-in
- Part 6: Installing the Cloud 9 WSAD/WSED Plug-in

There are two Java build solutions included with Cloud 9: the Java build for USS and the Java build for FTP (remote build and deploy). Each of the Java build solutions have their own sets of Translator control files. Those used in the USS Java build are described in Part 2 and those used for the FTP Java build are described in Part 3. The actual translators included with the product (CLZTJAVA, CLZTJAR and CLZTJBIZ) are supplied with both the USS and RBD control files in place, with the latter commented out. Within each installation desription, the samples always assume LANG=JAVA. This implies that either the USS build or the Remote FTP build is being configured.

If you want to make use of both the USS build and the FTP build, you can create separate translators for both the USS build and the FTP build and use different language types for both. For example: instead of using LANG=JAVA in the CLZTJAVA translator, set up two translators, one of which specifies LANG=JAVAUSS and the other specifies LANG=JAVAFTP.

TCP/IP considerations

When you are setting up the Cloud 9 server, you also need to consider your site's installation of TCP/IP. Cloud 9 uses an HTTP server and, therefore, needs to have the TCPIP.DATA file available to it. The UNIX System Services Planning Guide documents where the system finds this file. However, if you use another method of defining the location of this file (such as the System Resolver), you need to add a //SYSTCPD DD card to your Cloud 9 server job.

A TCP port needs to be available and it is good practice to reserve the port number.

To understand more about the System Resolver and TCPIP.DATA, see the following publications:

- z/OS UNIX Systems Services Planning
- z/OS IBM Communications Server: IP Configuration Guide

SMP/E installation

This manual does not cover the implementation aspects of Cloud 9. Rather, it is intended to guide the installer through a successful configuration of the major components.

The manual assumes that the System Modification Program/Extended (SMP/E) installation of Cloud 9 has been completed. The SMP/E instructions for Cloud 9 are in the *IBM Cloud 9 for SCLM for z/OS Program Directory*, GI10–3199.

Before you begin the Cloud 9 installation, take note that the following actions were recommended for the SMP/E installation::

- The root HFS file system was made read only
- The Cloud 9 directory was set up as a separate file system, mounted onto the root file system at /usr/lpp/Cloud9 (or whatever name you chose for your Cloud 9 root directory).

These recommendations conform with those specified in the *UNIX System Services Planning Guide* (GA22-7800-02). See the section headed "Deciding How to Mount Your Root HFS for Execution" for full details.

Separate SCLM installation

This manual does not cover the implementation and loading of the SCLM product, which is the source of data to the Cloud 9 interface.

Part 1. Cloud 9 base product installation

Chapter 1. Cloud 9 installation overview

This part of the manual contains the installation procedure for the Cloud 9 base product. The steps are organized into four major sections:

- Before you begin
- Create product initialization module
- Configure USS and HTTP server components
- Perform Installation Verification Procedures (IVP).

Product components used during installation

The following JCL, Parameter and HTML members are modified during the installation process. The names are provided here as an overview of naming standards and component functionality.

JCL members modified during installation

The following JCL members, located in SCLZJCL and resident on the host, are modified during installation:

CLZC9SRV JCL for the HTTP server task

CLZJUNIX JCL to copy members to Cloud 9 rootdir

CLZCHMOD REXX input to CLZJUNIX to reset permissions on files copied to rootdir in CLZJUNIX

CLZC9J01 JCL to build and initialize Suite Long Name Registry (SLR) database

CLZC9J03 JCL to create and initialize empty SLR database

CLZC9J04 JCL for SLR backup, delete, and define

CLZC9J05 JCL for standalone VSAM index expansion

CLZC9J06 JCL for SLR Installation Verification Procedure (IVP)

CLZC9JS4 JCL to build the CIGINI file

CLZJMIG JCL shell for Endevor conversion

CLZJIBM JCL shell for SCLM actions

CLZJDYN REXX shell for SCLM dynamic allocations

CLZC9TST JCL for environment diagnostic tests

HTTP server parameters modified during installation

The following HTTP Server parameters are copied from SCLZHTML to the Cloud 9 rootdir, where you modify them.

CLZHTTPD Sample HTTP server **httpd.conf** file

CLZEVARS Sample HTTP server **httpd.envvars** file

A step-by-step approach

Table 3. Installation steps for the Cloud 9 base product

Before you begin	
1.	Review system, software, and hardware requirements.
2.	Record site-specific information.
CP1.	Verify steps as shown in "Checkpoint #1 for Cloud 9 base product installation" on page 9.
Create product initialization module	
3.	Allocate and initialize SLR Long Name Registry database
4.	Set up the CIGINI initialization file
5.	Run the environment diagnostic tests (CLZC9TST).
CP2.	Verify steps as shown in "Checkpoint #2 for Cloud 9 base product installation" on page 18.
Configure USS and HTTP Server components	
6.	Modify the CLZHTTPD configuration member (rules file)
7.	Modify the CLZEVARS configuration member (environment variable)
8.	Customize the Cloud 9 HTTP Server JCL and supporting control files
9.	Create and populate additional HFS Cloud 9 directories
10.	Review authorization requirements for CLZRSDRV
CP3.	Verify steps as shown in "Checkpoint #3 for Cloud 9 base product installation" on page 37
Perform Installation Verification Procedures	
11.	HTTP Server Invocation IVP
CP4.	Verify steps as shown in "Checkpoint #4 for Cloud 9 base product installation" on page 40.
12.	Cloud 9 invocation and logon IVP
13.	Profile setup IVP.
14.	Batch and interactive IVPs
15.	SLR batch IVP.
16.	Set up the backup, delete, and define JCL for the SLR
CP5.	Verify steps as shown in "CHECKPOINT #5 for Cloud 9 Installation" on page 49

Chapter 2. Before you begin

Step 1: Review software and hardware requirements

In this step, you review the system, software, and hardware requirements for product installation.

System requirements

To successfully install Cloud 9, the following system requirements must be in place at your installation:

Table 4. System requirements for Cloud 9 installation

z/OS Operating System	Version 2 Release 7 (or higher)
IP address	Numerical IP address of host or named server on host
Port number	1024 or higher*
Product Load Library	Must be an APF authorized load library
Web browser	Microsoft Internet Explorer 5.0 or higher Netscape Navigator 4.7 to 6.x
*This port number must be higher than 1024, as port numbers lower than this are reserved for internal system services.	

Software requirements

Cloud 9 requires that SCLM be implemented for at least one project on the z/OS. Contact your system administrator to ensure that these requirements are in place.

VSAM exclusion

This product must be excluded from all VSAM buffering products. This must be done on a global basis. Failure to exclude SCLM Suite databases might result in damage to your files.

The VSAM buffering tool alters the buffers of the Cloud 9 SLR database. If a VSAM buffering product is active when adding files from the PC to SCLM (or during any add that needs to update the SLR), a problem might manifest itself in a number of ways:

- An abend might occur in the server job, with the following write to operator (wto) message:

```
CIG ABENDED THIS TASK DUE TO
    THE VSAM BUFFERS BEING ALTERED
    OR GLOBAL/LOCAL SHARED RESOURCES BEING USED
    BY A FOREIGN VSAM BUFFERING PRODUCT
    OR BY ADDING AMP PARMS IN THE JOB STREAM JCL
```

- The browser might stop loading and return the "Page contains no data" message.
- Abend U0007 might occur in the $$VSAM program.

Internationalization support

Text data transferred from the PC to Cloud 9 running under the IBM HTTP Server is translated from ASCII to EBCDIC using the TCP/IP ASCII-to-EBCDIC data conversion program, EZACJC04. This program is described in *TCP/IP V3R2 for MVS: API Reference* (SC31-7187-03). Data conversion is not performed on binary data. Determination of text versus binary data is made by the httpd.conf file.

Data sent from Cloud 9 running under the IBM HTTP Server to the PC is translated based on the MIME definition as specified in the httpd.conf file.

MIME implementation as used by the IBM HTTP Server is described in *OS/390 e-business Infrastructure: IBM HTTP Server 5.1 - Customization and Usage* (SG24-5603-00).

Step 2: Record site–specific information

Site–Specific Placeholders

The following placeholders represent values that are customer-specific.

- *dvolser*
- *dunit*
- *tdisk*
- *ispfqual*
- *password*
- *WEBJOBNAME*
- *user1*
- *user2*
- *portno*
- *rootdir*
- *ip-address*

These placeholders (see "Cloud 9 installation worksheet" on page 7 for definitions) are indicated in this chapter by the use of lowercase italics in the reproduced JCL. Substitute your site-specific values in all installation and implementation JCL. The value for placeholder "password" is provided for you, it is the word ***password***. The value for "password" is case insensitive. Complete the third column on the Placeholder worksheet for easy reference during installation.

ISPF/SCLM data set names

Additionally, identify the data set names for your current Interactive System Productivity Facility (ISPF) data sets as per the "Data set worksheet" on page 7. The current ISPF data set names are needed for the symbolic procedures used in batch.

Cloud 9 installation worksheet

Throughout the rest of the Cloud 9 installation process, you are asked to supply site-specific values for JCL and parameter modifications. You can use the following worksheet to record these values in one place for easy reference.

Table 5. Placeholder Worksheet

Place Holder	Definition	Your Site Value
dvolser	Volume serial number of the disk used to store permanent data sets (if needed).	
dunit	Unit label for permanent disk data sets (typically SYSDA). This specification is limited to 6 characters.	
tdisk	Unit label for temporary disk data sets (usually SYSDA). This specification is limited to 6 characters.	
ispfqual	High-level qualifier for the standard ISPF data sets.	
password		**password**
WEBJOBNAME	Job name of the HTTP server task. Also used in the ***httpd.conf*** file. This must be a value in upper-case.	
user1	User ID 1 for building IVP profile members	
user2	User ID 2 for building IVP profile members	
portno	TCP/IP port number for Cloud 9 invocation	
rootdir	Root directory for Cloud 9 HTTP Server See "SMP/E UNIX considerations" on page 8 for more information.	Through the SMP/E installation this value is set to **/usr/lpp/Cloud9/**
ip-address	Internet Protocol address for the HTTP server	

Data set worksheet

Cloud 9 relies on SCLM services to perform many of the Cloud 9 request functions. SCLM requires a proper ISPF environment to be established. This means that the Cloud 9 JCL and dynamic allocation routines must have the actual names of the ISPF data set names used at your installation. Use the following table to record the names of the actual ISPF data sets used at your installation. These are needed during "Modify batch shells" on page 25, where you modify the various USS JCL shell files.

Table 6. Data Set Worksheet

DDNAME	ISPF Data Set Examples	Your ISPF Data Set Names
SYSPROC	ISP.SISPCLIB	
ISPMLIB	ISP.SISPMENU	
ISPPLIB	ISP.SISPPENU	
ISPSLIB	ISP.SISPSLIB ISP.SISPENU	

SMP/E UNIX considerations

During the Cloud 9 SMP/E installion, UNIX directories, based on a ***PathPrefix*** variable, were created and populated. If the default values are used by your installation, your **rootdir** value is equal to:

```
/usr/lpp/Cloud9/
```

Your system administrator can provide more information.

Checkpoint #1 for Cloud 9 base product installation

At this point, the following libraries should have been allocated and populated. Using ISPF Option 3.4, verify that these files have been created and contain data.

Table 7. Checkpoint #1 for Cloud 9 Installation

Default Data Set Names	Your Data Set Names	Completed?
CLZ.SCLZDMDB		
CLZ.SCLZHTML		
CLZ.SCLZJCL		
CLZ.SCLZLOAD		
CLZ.SCLZPRF		
CLZ.SCLZCGI		
CLZ.SCLZPDF		
CLZ.SCLZJPG		

Before you begin: Cloud 9 base product installation

Chapter 3. Create Product Initialization Module

Step 3: Allocate and initialize an SLR database

In this step, you allocate and initialize the Suite Long Name Registry (SLR) long name support database. Long name support helps when moving, viewing and referencing objects from one platform to another. Many non-host objects have names greater than 8 characters, including the extension, and some of them are case sensitive.

The SLR database is where the correlation between the distributed platform object name and the standard host OS eight-character name is maintained. It is referenced in the CIGINI file.

At this point of the installation, the file is used in the IVP process.

Modify and submit CLZC9J01

To create this database, perform the following tasks:

1. Using ISPF EDIT, access member CLZC9J01 in the SCLZJCL data set.
2. Copy your job card values to the top of the member.
3. Substitute your site-specific values (identified on the "Cloud 9 installation worksheet" on page 7), as per the instructions in the comment area of the JCL.
4. Submit the job.

Note: The first time this job is run, it should end with COND CODE=8 for the delete function. If it does not:

1. Review your job card parameters and the JCL for errors.
2. Resubmit the job.

On subsequent runs, the job should end with COND CODE=0.

```
//**(JOBCARD)
//**
//*********************************************************************
//*                                                                   *
//* CLZC9J01 - THE PURPOSE OF THIS JCL IS TO CREATE A SLR DATABASE    *
//*            FOR IVP PURPOSES.                                      *
//*********************************************************************
//*                                                                   *
//* REQUIRED JCL MODIFICATION:                                        *
//*   1) INCLUDE A JOBCARD                                            *
//*   2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.      *
//*      - VOLUMES(DVOLSER)                                           *
//*                                                                   *
//* THE FOLLOWING MAY NOT REQUIRED FOR SMS INSTALLATIONS:             *
//*   - VOLUMES(DVOLSER)                                              *
//*                                                                   *
```

Figure 1. CLZC9J01 JCL (Part 1 of 2)

```
//*********************************************************************
//*                                                                   *
//* DO NOT MODIFY THE VSAM PARAMETERS PROVIDED IN THIS JCL. DOING SO *
//* WILL PRODUCE UNEXPECTED RESULTS FROM THE CLOUD9 APPLICATION.     *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//*  STEP 1: ALLOCATE THE SLR VSAM DATABASE AND REPRO THE RECORDS    *
//*          FROM THE INDD01 FILE.                                    *
//*                                                                   *
//*********************************************************************
//STEP1   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INDD01   DD DSN=CLZ.SCLZDMDB(CLZDEMO),DISP=SHR
//SYSIN   DD *
 DELETE CLZ.SCLZSLR.DATABASE
 DEFINE CLUSTER -
    (NAME('CLZ.SCLZSLR.DATABASE') -
    IMBED SPEED UNIQUE FREESPACE(30 30) -
    VOLUMES(DVOLSER) TRACKS(60 40) -
    SHR(4 3) -
    KEYS(254 0) -
    RECORDSIZE(512 1024)) -
  DATA  (CISZ(16000)) -
  INDEX (CISZ(4096))
 REPRO INFILE(INDD01) OUTDATASET('CLZ.SCLZSLR.DATABASE')
/*
```

Figure 1. CLZC9J01 JCL (Part 2 of 2)

Step 4: Set up the CIGINI initialization file

In this step, you create the CIGINI member, a file in text format (contains data only, no executable code) that contains various product parameters such as product password, database names, and the product load library name. For test purposes, you create a new version of this file.

Modify and submit CLZC9JS4

The CIGINI load module must be located in the Cloud 9 steplib or linklist area. Create the CIGINI load module by executing the JCL in member CLZC9JS4 of the SCLZJCL data set. As input to the job, you need to:

1. Using ISPF EDIT, access member CLZC9JS4 in the SCLZJCL data set.
2. Copy your job card values to the top of the member.
3. Substitute your site-specific values (identified on the “Cloud 9 installation worksheet” on page 7) as per the instructions in the comment area of the JCL.

   Note: If the Cloud 9 Java component is to be used in the same SCLM project as Breeze, then the keywords for the Breeze CIGINI generation need to be included here. For more information about the IBM Breeze for SCLM for z/OS product, see the *Breeze Installation Guide*, SC31–8819.
4. Update Cloud 9 password.
5. Verify that the SYSLMOD points to the intended execution library.
6. Submit the job.

Note: This job should end with COND CODE=0. If it does not:

1. Review your job card parameters and the JCL for errors.
2. Resubmit the job.

```
//* (JOBCARD)
//* ------------------------------------------------------------------*
//*  NAME:    CLZC9JS4                                                 *
//*  PURPOSE: PARSE, COMPILE AND LINK THE CIGINI MODULE.               *
//*                                                                    *
//* ------------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                        *
//*                                                                    *
//*     1.     PERFORM MODIFICATION ON THE CIGINI STATEMENTS.          *
//*            THIS SAMPLE WILL NOT COMPILE AS DELIVERED.              *
//*     2.     INSERT A VALID JOB CARD WITH A VALID CLASS              *
//*     3.     CHANGE THE UNIT=TDISK TO THE APPROPRIATE UNIT           *
//*            NAME FOR TEMPORARY FILES.                               *
//*     4.     MAKE SURE THE SYSLMOD POINTS TO THE INTENDED            *
//*            EXECUTION LIBRARY.                                      *
//*                                                                    *
//* 11NOV2001 RMCC - APAR OW52105 CHANGES FOR BREEZE CO-EXISTENCE      *
//*                                                                    *
//*--------------------------------------------------------------------*
//*                                                                    *
//* STEP 1: PARSE CIGINI SYNTAX.  BUILD INPUT FOR ASSEMBLER.           *
//*                                                                    *
//*--------------------------------------------------------------------*
//PARSE    EXEC PGM=CLZMPILE
//STEPLIB  DD  DSN=CLZ.SCLZLOAD,DISP=SHR  <--- CLOUD 9 LIBRARY
//CIGIN    DD  *
* ------------------------------------------------------------------ *
*   .   COMMON SECTION                                               *
*                                                                    *
*       PLEASE MODIFY TO MEET YOUR NAMING STANDARDS.                 *
*                                                                    *
*     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  *
*     ! NOTE: THERE ARE TWO PRODUCT LOADLIB STATEMENTS IN         !  *
*     ! THE INPUT. THIS IS BECAUSE, THE CLOUD9 SERVER REQUIRES    !  *
*     ! AN AUTHORISED LOADLIB. IF THE DATASET USED IN THE SERVER  !  *
*     ! JCL IS DIFFERENT THAN THE INSTALL LIBRARY, THE CIGINI     !  *
*     ! WILL HAVE TO BE ASSEMBLED POINTING TO THE AUTHORISED      !  *
*     ! LIBRARY.                                                  !  *
*     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  *
*                                                                    *
* ------------------------------------------------------------------ *
DEFINE COMMON SECTION
  PRODUCT LOADLIB     = 'CLZ.SCLZLOAD'
* PRODUCT LOADLIB     = 'CLZ.SCLZLOAD'
  WORK UNIT           = TDISK
  VIO  UNIT           = TDISK
  DO NOT ALLOW ALTERNATE CIGINI FILE
```

Figure 2. CLZC9JS4 JCL and Input (Part 1 of 3)

```
* ------------------------------------------------------------------ *
*   .   BREEZE INPUT TO COMMON SECTION                               *
*                                                                    *
*       THIS IS BREEZE SPECIFIC CIGINI INPUT,                        *
*       PLEASE MODIFY TO MEET YOUR NAMING STANDARDS.                 *
*       IF YOU ARE NOT USING BREEZE THEN REMOVE THESE                *
*       TWO STATEMENTS BELOW.                                        *
*                                                                    *
* ------------------------------------------------------------------ *

  JAVASERVERCONTROL DSNAME = 'BZZ.SBZZJAVA'
                    MEMBER =  BZZ$CNTL

* ------------------------------------------------------------------ *
*   .   CLOUD9 SECTION                                               *
*                                                                    *
*       THIS IS CLOUD 9 SPECIFIC CIGINI INPUT,                       *
*       PLEASE MODIFY TO MEET YOUR NAMING STANDARDS.                 *
*                                                                    *
* ------------------------------------------------------------------ *
DEFINE CLOUD9 SECTION
  PASSWORD = 'PASSWORD'
  SLRVSAM DSNAME = 'CLZ.SCLZSLR.DATABASE'
* ENDEVORBRIDGE

* ------------------------------------------------------------------ *
*   .   BREEZE BRSCLM SECTION                                        *
*                                                                    *
*       THIS IS BREEZE SPECIFIC CIGINI INPUT,                        *
*       PLEASE MODIFY TO MEET YOUR NAMING STANDARDS.                 *
*       IF YOU ARE NOT USING BREEZE THEN REMOVE THIS SECTION.        *
*                                                                    *
* ------------------------------------------------------------------ *
DEFINE BRSCLM  SECTION
  PASSWORD = 'PASSWORD'
  VSAM DSNAME = 'BZZ.SBZZPKG.DATABASE'
/*
//CIGPUNCH DD  DSN=&&TEMP,DISP=(NEW,PASS),
//            UNIT=TDISK,SPACE=(10,10),
//            DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
```

Figure 2. CLZC9JS4 JCL and Input (Part 2 of 3)

```
//*-------------------------------------------------------------------*
//*                                                                   *
//* STEP 2: ASSEMBLE THE CIGINI INPUT CREATED IN STEP 1.              *
//*                                                                   *
//*  NOTE: CHOOSE THE DESTINATION OF YOUR CIGINI MODULE.              *
//*                                                                   *
//*-------------------------------------------------------------------*
//ASM      EXEC PGM=ASMA90,
//             REGION=3072K,
//             COND=(0,NE),
//             PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT)'
//SYSIN    DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSLIN   DD  DSN=&&SYSLIN,
//             UNIT=TDISK,SPACE=(TRK,(3,5)),
//             DISP=(NEW,PASS,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
//SYSPRINT DD  SYSOUT=*
//*-------------------------------------------------------------------*
//*                                                                   *
//* STEP 3: LINK EDIT THE CIGINI MODULE                               *
//*                                                                   *
//*  NOTE: CHOOSE THE DESTINATION OF YOUR CIGINI MODULE. IF YOU ARE   *
//*        PLANNING ON USING AN ALTERNATE CIGINI MODULE, YOU MUST     *
//*        FIRST BUILD A CIGINI THAT RESIDES IN A STEPLIB LIBRARY.    *
//*-------------------------------------------------------------------*
//LINK     EXEC PGM=IEWL,
//             REGION=2048K,
//             PARM='LIST,NCAL,XREF,LET,RENT,REUS',
//             COND=(0,NE)
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&SYSLIN,
//             DISP=(OLD,DELETE,DELETE)
//SYSLMOD  DD  DSN=CLZ.SCLZLOAD(CIGINI),   <--- LOCATION OF CIGINI MODUL
//             DISP=SHR
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
```

Figure 2. CLZC9JS4 JCL and Input (Part 3 of 3)

Define COMMON section

This section is always required. The COMMON section describes parameters required by all products.

Table 8. CLZC9JS4 - Define COMMON section

Syntax	Purpose	Usage
PRODUCT LOADLIB = *CLZ.SCLZLOAD*	Defines the name of the product load library. Default: None	Required.
WORK UNIT = tdisk	Defines DASD unit name for temporary disk files. Default: None	Required.
VIO UNIT = tdisk	Defines DASD unit name for temporary disk files in those situations where the product can take advantage of VIO disk access.	Required.

Table 8. CLZC9JS4 - Define COMMON section (continued)

Syntax	Purpose	Usage
JAVASERVERCONTROL DSNAME = 'BZZ.SBZZJAVA' MEMBER = BZZ$CNTL	Defines the data set and member that contains the Java control data sets required by Breeze. Substitute the dsname parameter with the file name at your location.	Required if Breeze for SCLM is also being installed.

Define Cloud 9 section

Table 9. CLZC9JS4 - Define Cloud 9 Section

Syntax	Purpose	Usage
PASSWORD = password	This required keyword and variable are checked during invocation of the product. For the IBM version of this product, specify PASSWORD=PASSWORD. Default: None	Required.
SLRVSAM DSNAME = 'CLZ.SCLZSLR.DATABASE'	This optional keyword and variable is checked when transferring files from and to the browser. The SLR is for supporting long names for distributed types.	Optional.
ENDEVORBRIDGE	This optional keyword is used when the user is converting from CA-Endevor to SCLM.	Optional.

Define Breeze Section

Table 10. CLZC9JS4 - Define Breeze section

Syntax	Purpose	Usage
PASSWORD = password	This keyword and variable are checked during invocation of the product. For the IBM version of this product, specify PASSWORD = 'PASSWORD'. Default: None	Required if Breeze for SCLM is being installed.
VSAM DSNAME = 'BZZ.SBZZPKG.DATABASE'	This keyword contains the name of the Breeze for SCLM VSAM database. Substitute the file name here with the file name in your location.	Required if Breeze for SCLM is being installed.

Step 5: Run environment diagnostic tests

The CLZC9TST is an installation verification program that performs three environmental diagnostic tests for Cloud 9. These tests are:

- Test that the target LOAD library is an APF-AUTHORIZED library.
- Check that access is possible to TCP/IP.
- Check if a REXX run-time library or an alternative REXX library is available.

Modify and submit CLZC9TST

To run the job, perform the following tasks:

1. Using ISPF EDIT, access the member in the SCLZJCL target library.
2. Copy your job card values to the top of the member
3. Substitute your site-specific values (identified on the "Cloud 9 installation worksheet" on page 7), as per the instructions in the comment area of the JCL member.
4. Submit the job.

Note: This job should end with COND CODE=0. If it does not:
1. Review your job card parameters and the JCL member for errors.
2. Resubmit the job

```
//**(JOBCARD)
//**
//*********************************************************************
//*                                                                   *
//* NAME     - CLZC9TST                                               *
//* PURPOSE  - THE PURPOSE OF THIS JCL IS TO RUN THREE ENVIRONMENTAL *
//*            DIAGNOSTICS FOR CLOUD 9.                               *
//*                                                                   *
//*            THIS JOB WILL :-                                       *
//*            1.   TEST THAT THE TARGET LOAD LIBRARY IS AN           *
//*                 APF-AUTHORIZED LIBRARY.                           *
//*            2.   CHECK THAT ACCESS IS POSSIBLE TO TCP/IP.          *
//*            3.   CHECK IF A REXX RUNTIME OR THE  REXX ALTERNATE    *
//*                 LIBRARY IS AVAILABLE.                             *
//*                                                                   *
//* 20NOV2001 RMCC - APAR OW52105 IMPROVE COMMENTS                    *
//*********************************************************************
//*                                                                   *
//* REQUIRED JCL MODIFICATION:                                        *
//* 1.   INCLUDE A JOBCARD                                            *
//* 2.   MAKE SURE THAT THE STEPLIB IS EXACTLY THE SAME AS THE ONE    *
//*      USED IN THE CLOUD 9 SERVER JOB.                              *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//*  STEP 1: PERFORM ENVIRONMENTAL TESTS.                             *
//*                                                                   *
//*********************************************************************
//STEP1   EXEC PGM=CLZATEST
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR
//         DD DSN=TCPIP.SEZATCP,DISP=SHR
//* DD DSN=REXX.V1R3M0.SEAGALT,DISP=SHR  <-- REXX ALTERNATE LIBRARY
//CIGRPT  DD SYSOUT=*
```

Figure 3. CLZC9TST JCL and Input

Checkpoint #2 for Cloud 9 base product installation

At this point the SLR VSAM database should have been created and populated and the CIGINI initialization module should have been created and stored in the product load library.

Table 11. Checkpoint #2 for Cloud 9 base product installation

Task	Completed?
Allocate and initialize the SLR database?	
Build a CIGINI file that points to the demo database?	
Run environment diagnotics IVP CLZC9TST?	

Chapter 4. Configure USS and HTTP Server components

Preparation

Before you begin the configuration of the HTTP Server parameters and JCL, it is important to review a few key topics.

Stand-alone HTTP Server job

Cloud 9 supplies some sample JCL that will run an IBM HTTP Server as a stand-alone job. Your installation might already have an IBM HTTP Server job running, and you might want to merge the Cloud 9 application into the active HTTP configuration. This can be done, but you should not attempt it without the cooperation of the HTTP Server administrator.

Sample HTTPD configuration files

The CLZHTTPD and CLZEVARS examples are set as default. They are minimally configured for Cloud 9 only. Before using them, you need to review them with your site's HTTP server administrator, and then modify them to ensure that they meet all of your installation's specific settings.

Additional information

There are two IBM manuals that can be of assistance when configuring your HTTP server:

- *HTTP Server Planning, Installing, and Using*
- *OS/390 e-business Infrastructure: IBM HTTP Server 5.1 — Customization and Usage* (This is an IBM Redbook)

For the publication order numbers for these books, see "Where to find more information" on page xi.

Step 6: Modify the CLZHTTPD configuration member (rules file)

In this step, you review and modify the CLZHTTPD member off–loaded from the product tape into the **CLZ.SCLZHTML** file. This member is named the ***rules file*** in HTTP server configuration terminology and is pointed to by the server JCL parameter list. Because many of the members contain case sensitive values, **issue the CAPS OFF command** to ensure that these are not automatically changed to uppercase during editing.

Review root directory and port number values

Figure 4 on page 20 shows only the lines that change in the CLZHTTPD ***rules file***, based on the *rootdir* and *portno*. Ask your HTTP server administrator to review this member.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       CLZ.SCLZHTML(CLZHTTPD) - 01.00               Columns 00001 00072
Command ===>                                              Scroll ===> CSR
****** **************************** Top of Data ******************************
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 19 Line(s) not Displayed
==CHG> ServerRoot     rootdir/
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  2 Line(s) not Displayed
==CHG> Port      portno
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  7 Line(s) not Displayed
==CHG> Protection WEBJOBNAME {
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 10 Line(s) not Displayed
==CHG> PidFile        rootdir/httpd-pid
==CHG> #AccessLog     rootdir/logs/httpd-log
==CHG> #AgentLog      rootdir/logs/agent-log
==CHG> #RefererLog    rootdir/logs/referer-log
==CHG> #ErrorLog      rootdir/logs/httpd-errors
==CHG> #CgiErrorLog   rootdir/logs/cgi-error
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 11 Line(s) not Displayed
==CHG> AccessReportRoot  rootdir/reports
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 48 Line(s) not Displayed
==CHG> Exec            /cgi-bin/*       rootdir/cgi-bin/*
==CHG> Pass            /html/*          rootdir/html/*
==CHG> Pass            /*               rootdir/*
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  193 Line(s) not Displayed
****** **************************** Bottom of Data ***************************
```

Figure 4. CLZHTTPD (httpd.conf) Change Fields Only

ADDTYPE directives

ADDTYPE directives are used to control the MIME types and file transfer defaults between the browser and the mainframe. Cloud 9 searches for ADDTYPE definitions in the following locations:

- The MVS file or USS file specified by the C9_ADDTYPE_FILE variable defined in httpd.envvars (see "Step 7: Modify the CLZEVARS Configuration Member (environment variable)" on page 21 for information about customising httpd.envvars)
- The USS file used as the RULE_FILE on the Cloud9 Web server job (i.e. the file specified with the -r flag on the PARM= statement) or, if this is not specified, the httpd.conf file in the /etc/ directory.

Consequently, if C9_ADDTYPE_FILE is not specified, Cloud 9 searches the RULE_FILE, and if the RULE_FILE is not specified on the server JCL, Cloud 9 uses the MIME types defined in the httpd.conf file in /etc/ directory.

Note: The file specified by C9_ADDTYPE_FILE can be a USS file or an MVS file.

The sample ***rules file*** also contains ADDTYPE directives that control the MIME commands and file transfer defaults between the browser and the mainframe. As the implementation continues, these ADDTYPES might need to be expanded to accommodate additional file types and requirements. At this point, there are no additional modifications required for the ADDTYPE definitions.

Modify and save CLZHTTPD

1. Review the *rootdir*, *portno*, and *WEBJOBNAME* variables from the Cloud 9 Installation Worksheet.
2. Using ISPF EDIT, access member CLZHTTPD in the CLZ.SCLZHTML data set.
3. Issue the CAPS OFF command to ensure case sensitive values do not become uppercased.
4. Issue the following global commands against the member:
   - X ALL
   - F rootdir ALL
   - F portno ALL
   - F WEBJOBNAME
   - Change rootdir ALL *rootdir* (ensure that the end format of the rootdir is **/rootdir/**)
   - Change portno ALL *portno*
   - Change WEBJOBNAME ALL *JOBNAME*
5. Save the member.

Step 7: Modify the CLZEVARS Configuration Member (environment variable)

In this step, you review and modify the CLZEVARS member that was off–loaded from the installation tape into the **CLZ.SCLZHTML** file. This member is named the ***environment variable file*** in HTTP server configuration terminology and is pointed to by the server JCL parameter list. Because many of the members contain case sensitive values, **issue the CAPS OFF command** to ensure that these are not automatically changed to uppercase during editing.

Review the CLZEVARS member

Figure 5 on page 22 shows the contents of the CLZEVARS member. This file must be configured by your HTTP server administrator, as many of the parameters are site-specific. Note that:

1. The *rootdir* variable from the Cloud 9 Installation Worksheet is needed.
2. The STEPLIB directive is delivered as STEPLIB=CURRENT. This means the HTTP server–spawned tasks default to the STEPLIB in the Cloud 9 server JCL. If your installation uses STEPLIB=dsn1,dsn2, all of the data sets in the STEPLIB statement must be authorized, as Cloud 9 requires an authorized environment.

```
#--------------------------------------------------------------------
#  Name:      C9EVARS   (Will be named  rootdir/httpd.envvars in UNIX.)
#  Purpose:   Cloud9 Server Environment variable parameters
#  Usage:     This file is pointed to in the CIGC9SRV JCL.
#--------------------------------------------------------------------
#To customize this file change:
#1. rootdir as per the Cloud9 installation worksheet.
#2. If required, include the C9_ADDTYPE_FILE variable and a USS file
#   or MVS file that specifies the mime types and translation rules in
#   the following locations:
#   a. MVS file or USS file specifid by the C9_ADDTYPE_FILE in
#      httpd.envvars (this member)
#   b. The USS file used as the RULE_FILE on the Cloud9 web server
#      job (i.e., the file specified with the -r option)
#   c. The httpd.conf file in the SERVER_ROOT dirctory.
#   Consequently, if C9_ADDTYPE_FILE is not specified, the Cloud9
#   will search the RULE_FILE, and if the RULE_FILE does not define
#   mime types, then Cloud9 will use the mime types defined in the
#   httpd.conf file in SERVER_ROOT.
#   Note that the file specified by C9_ADDTYPE_FILE can be a USS file
#   or a MVS file.
#
#Various install and configuration paths are currently set /usr/...
#  This and all other parms using /usr/ must be reviewed with
#  the HTTP Server administrator as these set up issues are global
#  in nature versus Cloud9 specific usage.
#--------------------------------------------------------------------
PATH=/bin:.:/usr/sbin:/usr/lpp/internet/bin:/usr/lpp/internet/sbin:
/usr/lpp/ldap/bin:rootdir/bin:<JAVA_HOME>/bin 1
SHELL=/bin/sh
TZ=EST5EDT
LANG=C
LC_ALL=en_US.IBM-1047
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:
/usr/lpp/ldap/lib/nls/msg/%L/%N 1
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/ldap/lib:
<JAVA_HOME>/lib/mvs/native_threads 1
JAVA_HOME=<JAVA_HOME>
CLASSPATH=.:/usr/lpp/internet/server_root/CAServlet:
<JAVA_HOME>/lib/classes.zip 1
STEPLIB=CURRENT
SERVER_ROOT=rootdir/
#C9_ADDTYPE_FILE=rootdir/c9addtype_filename
```

Figure 5. CLZEVARS (httpd.envvars) Sample Member

1 These paths have been split over two lines for display purposes only. They exist as one line in your sample member.

Modify and save CLZEVARS

Now that you have reviewed the considerations and contents of the CLZEVARS file, use the following instructions to customize the global variables and local variables.

1. Review the *rootdir* variable from the Cloud 9 Installation Worksheet.
2. Using ISPF EDIT, access member CLZEVARS in the CLZ.SCLZHTML data set you off-loaded from the installation tape.
3. Issue the CAPS OFF command to ensure that case sensitive values are not changed to upper case.
4. Perform the following global commands against the member:
   - Change rootdir ALL *rootdir*

5. If you wist to use the C9_ADDTYPE_file mentioned in "ADDTYPE directives" on page 20, then un-comment the variable and enter a file name that contains the addtype directives.
6. Change any of the other local directory settings as per the HTTP server administrator's direction.
7. Save the member.

Step 8: Customize the Cloud 9 HTTP Server JCL and supporting control files

Copy Product load library into authorized library

The Cloud 9 server must run from an authorized library, included in an authorized steplib concatenation. If the product load library is not an authorized data set, then you must copy the product load library into the authorized library for server execution.

Effects on other JCL

WARNING: If the authorized library name has changed from the installed library, make sure you review your application JCL members **CLZJIBM**, **CLZJMIG** and **CLZJDYN**, for possible steplib changes.

Modify CLZC9SRV

The CLZC9SRV member contains JCL that invokes the Cloud 9 HTTP server. It uses many default HTTP settings that can be modified and tailored by your HTTP server administrator. For the initial installation, invoke the server "**as is**" and then customize it to your needs at a later date.

Timeout parameter

WARNING: This job must not time out. Do not remove the TIME=NOLIMIT parameter on the EXEC statement. This job can also be made a started task.

HTTP Server, owner user ID considerations

Whether the Cloud 9 HTTP Server job is to be submitted as a started task or as a job, you need to define to RACF (or other security product) an OMVS segment that includes a UID and home directory for the user ID associated with the HTTP Server.

Modify and submit CLZC9SRV

To start the Cloud 9 server, perform the following tasks:

1. Using ISPF EDIT, access member CLZC9SRV in the CLZ.SCLZJCL data set you off-loaded from the installation tape.
2. Issue the CAPS OFF command to ensure that case sensitive values are not changed to uppercase.
3. Copy a UNIX Supervisor-Level job card with password to the top of the member. This jobcard **REQUIRES** a user ID and password with enough authority to load the HTTP server application. If the user ID authority is not sufficient, the server task ends with an 'insufficient authority' message on the console.
4. Change Jobname to equal WEBJOBNAME. Use the value from the "Cloud 9 installation worksheet" on page 7, which *must* match the WEBJOBNAME in the CLZHTTDP member or rootdir/httpd.conf file. **A sample job card is provided in the JCL member in Figure 6 on page 24.**

5. Substitute your site-specific values (identified on the "Cloud 9 installation worksheet" on page 7) as per the instructions in the comment area of the JCL.
6. Save the member (Do not submit this job).

```
//* (Sample Jobcard)
//*
//*WEBJOBNAME JOB (ACCT#),'COMMENT',CLASS=A,REGION=0M,
//*    MSGCLASS=H,MSGLEVEL=(1,1),USER=XXXX,PASSWORD=XXXXXXX
//*
//*  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//*  ! This jobcard REQUIRES the userid and password with enough   !
//*  ! authority to load the HTTP application.  If the userid      !
//*  ! authority is not sufficient, then the server task will end  !
//*  ! with an 'insufficient authority' message on the console.    !
//*  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//*  ! WEBJOBNAME - The value from the placeholder worksheet.      !
//*  !  (The server job name must match the WEBJOBNAME             !
//*  !  in the c9httdp member or rootdir/httpd.conf file.)         !
//*  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//*
//*-------------------------------------------------------------------
//* THIS IS THE CLOUD 9 FOR SCLM DEFAULT IBM HTTP WEB SERVER JCL
//*-------------------------------------------------------------------
//*
//* INSTRUCTIONS:
//*   1. CHANGE ROOTDIR TO THE VALUE IN THE CLOUD9 WORKSHEET.
//*   2. CHANGE PORTNO TO THE VALUE IN THE CLOUD9 WORKSHEET.
//*   3. CHANGE WEBJOBNAME TO VALUE IN THE CLOUD9 WORKSHEET.
//*   4. BE CAREFUL TO USE THE PROPER CASE WHEN CHANGING VALUES.
//*   5. IF THE CLZ.SCLZLOAD IS NOT AUTHORIZED, THEN
//*      COPY CURRENT CONTENTS OF CLZ.SCLZLOAD INTO EXISTING
//*      AUTHORIZED DATASET OR GET CLZ.SCLZLOAD AUTHORIZED.
//*   6. REVIEW THE NAME OF THE TCPIP LIBRARY FOR SITE
//*      STANDARDS. ACCESS TO THIS LIBRARY IS REQUIRED FOR
//*      CLOUD 9.  YOU WILL NOT NEED TO INCLUDE THE TCPIP LIBRARY
//*      IF IT IS IN THE LINKLIST.
//*   7. REVIEW THE httpd.envvars CONFIGURATION FILE FOR A 'STEPLIB'
//*      STATEMENT. IF THERE IS A STEPLIB= STATEMENT THAT INCLUDES
//*      DATASET NAMES, THEN ENSURE THAT THIS LIST IS AUTHORIZED.
//*   8. AFTER CHANGING THE ROOTDIR AND PORT VALUES, REVIEW THE
//*      EXECUTION PARM. THE PARM STRING SHOULD GO UP THROUGH COL 71
//*      AND THEN CONTINUE IN COL 16 ON THE NEXT LINE.
//*   9. IF YOU ARE A BREEZE USER THEN REVIEW THE NAME OF THE BREEZE
//*      LOAD LIBRARY, AND UNCOMMENT THE LINE LABELED 'BREEZE USERS'.
//*
//*-------------------------------------------------------------------
//*  The parm variable on the EXEC statement is of the format:
//*  (LEPARMS/ICSPARMS).
//*
//*  Refer to the following manuals for more information:
//* 1. HTTP Server Planning,Installing, and Using SC31-8690-02
//* 2. Redbook:OS/390 e-business Infrastructure: IBM HTTP Server 5.1
//*   - Customization and Usage  SG24-5603-00
//*-------------------------------------------------------------------
//CIGWEB  EXEC PGM=IMWHTTPD,TIME=NOLIMIT,
//     ACCT=(ACCT#),
//     PARM=('ENVAR("_CEE_ENVFILE=rootdir/httpd.envvars")/-r rootdir/
//             httpd.conf -B -p portno')
```

Figure 6. CLZC9SRV (Part 1 of 2)

```
//* ------------------------------------------------------------- *
//* This JCL requires an authorized dataset.  Review instruction  *
//*  numbers 5-7 above.                                           *
//* ------------------------------------------------------------- *
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR
//*        DD DSN=BZZ.SBZZLOAD,DISP=SHR   * BREEZE USERS uncomment  *
//*        DD DSN=TCPIP.SEZATCP,DISP=SHR  * Uncomment if not in     *
//*                                       * LINKLIST or LPA         *
//SYSIN    DD DUMMY
//OUTDSC   OUTPUT DEST=HOLD
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//CEEDUMP  DD SYSOUT=*,OUTPUT=(*.OUTDSC)
```

Figure 6. CLZC9SRV (Part 2 of 2)

Modify batch shells

Modify CLZJIBM

1. Using ISPF EDIT, access member CLZJIBM in the installed CLZ.SCLZJCL data set.
2. Skip the step of adding your job card values. The jobcard is provided from the job card information in your Cloud 9 profile (see “Step 14: Batch and Interactive IVPs” on page 42).
3. Substitute your site-specific values (identified on the “Cloud 9 installation worksheet” on page 7) as per the instructions in the comment area of the JCL.
4. Substitute the ISPF data set names for your installation.

   Note: This member needs to use the same names as the customized FLMLIBS skeleton member for SCLM.
5. Save the member.

Figure 7 on page 26 shows the CLZJIBM batch JCL member.

```
)DOT
%JOBCARD%
)ENDDOT
//* ------------------------------------------------------------------ *
//* NAME:     CLZJIBM                                                  *
//* PURPOSE:  CLOUD 9 FOR SCLM.                                        *
//*           SCLM BATCH SKELETON.                                     *
//* ------------------------------------------------------------------ *
//*                                                                    *
//* REQUIRED JCL MODIFICATION:                                         *
//*  1) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.        *
//*      - ISPFQUAL                                                    *
//*      - TDISK                                                       *
//*  2) TARGET LIBRARIES CLZ.SCLZLOAD AND CLZ.SCLZCGI        /* C1 */ *
//*     SHOULD BE CHANGED IF THE INSTALLED HIGH LEVEL        /* C1 */ *
//*     QUALIFIER IS NOT 'CLZ.'                              /* C1 */ *
//*                                                                    *
//*  NOTE: BREEZE USERS MUST MODIFY THIS MODULE PER EMBEDDED           *
//*        INSTRUCTION. BREEZE DATASET NAMES MAY ALSO NEED TO BE       *
//*        MODIFIED.                                                   *
//*                                                                    *
//* 22OCT2001 OW51810 - CHANGES MARKED AS /* C1 */                     *
//* Z020402A                                                           *
//* Z240109A  M344 - AUTHCODE PROCESSING                               *
//*                                                                    *
//*--------------------------------------------------------------------*
//*  RESIDES IN HTTP SERVER AT:                          /* C1 */ *
//*  /ROOTDIR/CLOUD9/JCL/CLZJIBM                                     *
//*--------------------------------------------------------------------*
)IF ACTION=OCOPY
//COPY   EXEC PGM=IKJEFT01
)DOT
%COPYFILES%
)ENDDOT
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
)DOT
%OCOPYSYNTAX%
)ENDDOT
/*
)ENDIF
```

Figure 7. CLZJIBM (Part 1 of 4)

```
)IF ACTION=IEBCOPY
//COPY     EXEC PGM=IEBCOPY
)DOT
%COPYFILES%
)ENDDOT
//SYSIN   DD *
)DOT
%OCOPYSYNTAX%
)ENDDOT
//SYSPRINT  DD  SYSOUT=*
)ENDIF
//*-------------------------------------------------------------------
//GENER  EXEC PGM=IEBGENER
//SYSUT1   DD *
)DOT
%SCLMSYNTAX%
)ENDDOT
//SYSUT2   DD DSN=&&CLIST(TEMPNAME),UNIT=TDISK,
//            SPACE=(TRK,(10,10,2),RLSE),
//            DISP=(NEW,PASS),DCB=(LRECL=80,
//            BLKSIZE=1600,DSORG=PO,RECFM=FB)
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
)IF ACTION=DELETE
//DGRPTS   DD DSN=&&DELLIST,DISP=(NEW,PASS),                DELETE
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
)IF ACTION=BUILD
//COPYBULD EXEC PGM=CLZTFILE                                JAVA
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR                        /* C1 */
//SYSIN    DD *                                             JAVA
)DOT
%SCLMSYNTAX%
)ENDDOT
//SYSOUT   DD DSN=&&BSYNTAX,DISP=(NEW,PASS),                JAVA
//            SPACE=(TRK,(10,10),RLSE),UNIT=TDISK,          JAVA
//            DCB=(LRECL=80,BLKSIZE=0,DSORG=PS,RECFM=FB)    JAVA
)ENDIF
//*****************************************************************
//* BREEZE USERS:  UNCOMMENT LINES WITH "BREEZE USERS"
//*****************************************************************
//TSO    EXEC PGM=IKJEFT01,REGION=4096K,TIME=1439,DYNAMNBR=200
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR
//*        DD DSN=BZZ.SBZZLOAD,DISP=SHR            BREEZE USERS
//SYSTSIN  DD *
  ISPSTART CMD(%TEMPNAME)
//SYSTSPRT DD SYSOUT=(*)
//SYSPROC  DD DSN=&&CLIST,DISP=(OLD,DELETE)
//*        DD DSN=BZZ.SBZZCLIB,DISP=SHR            BREEZE USERS
//*****************************************************************
```

Figure 7. CLZJIBM (Part 2 of 4)

```
//* ISPF LIBRARIES
//ISPMLIB  DD DSN=ISPFQUAL.SISPMENU,DISP=SHR
//*        DD DSN=BZZ.SBZZMENU,DISP=SHR            BREEZE USERS
//ISPSLIB  DD DSN=ISPFQUAL.SISPSENU,DISP=SHR
//         DD DSN=ISPFQUAL.SISPSLIB,DISP=SHR
//*        DD DSN=BZZ.SBZZSENU,DISP=SHR            BREEZE USERS
//ISPPLIB  DD DSN=ISPFQUAL.SISPPENU,DISP=SHR
//*        DD DSN=BZZ.SBZZPENU,DISP=SHR            BREEZE USERS
//ISPTLIB  DD UNIT=VIO,DISP=(NEW,PASS),SPACE=(CYL,(1,1,5)),
//            DCB=(LRECL=80,BLKSIZE=19040,DSORG=PO,RECFM=FB)
//         DD DSN=ISPFQUAL.SISPTENU,DISP=SHR
//ISPTABL  DD UNIT=VIO,DISP=(NEW,PASS),SPACE=(CYL,(1,1,5)),
//            DCB=(LRECL=80,BLKSIZE=19040,DSORG=PO,RECFM=FB)
//ISPPROF  DD UNIT=VIO,DISP=(NEW,PASS),SPACE=(CYL,(1,1,5)),
//            DCB=(LRECL=80,BLKSIZE=19040,DSORG=PO,RECFM=FB)
//ISPLOG   DD SYSOUT=*,
//            DCB=(LRECL=120,BLKSIZE=2400,DSORG=PS,RECFM=FB)
//ISPCTL1  DD DISP=NEW,UNIT=VIO,SPACE=(CYL,(1,1)),
//            DCB=(LRECL=80,BLKSIZE=800,RECFM=FB)   TEMPORARY FILE
//ZFLMDD   DD *
    ZFLMNLST=FLMNLENU  ZFLMTRMT=ISR3278   ZDATEF=YY/MM/DD
/*
//*********************************************************************
//* SCLM OUTPUT FILES
//*********************************************************************
//FLMMSGS  DD SYSOUT=(*)
)IF ACTION=BUILD
//BLDMSGS  DD SYSOUT=*,                                      BUILD
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//BLDREPT  DD SYSOUT=*,                                      BUILD
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FBA)
//BLDLIST  DD SYSOUT=*,                                      BUILD
//         DCB=(LRECL=259,BLKSIZE=3120,RECFM=VB)
//BLDEXIT  DD DSN=&&BLDEXIT,DISP=(NEW,DELETE),               BUILD
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=160,BLKSIZE=3200,RECFM=FB)
//BSYNTAX  DD DSN=&&BSYNTAX,DISP=(OLD,PASS)                  JAVA
)ENDIF
)IF ACTION=PROMOTE
//PROMMSGS DD SYSOUT=*,                                      PROMOTE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=FB,DSORG=PS)
//PROMREPT DD SYSOUT=*,                                      PROMOTE
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB,DSORG=PS)
//PROMEXIT DD DSN=&&PROMEXIT,DISP=(NEW,DELETE),              PROMOTE
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=160,BLKSIZE=3200,RECFM=FB)
//COPYERR  DD SYSOUT=*,                                      PROMOTE
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
)ENDIF
)IF ACTION=MIGRATE
//U2LSTS   DD SYSOUT=*,                                      MIGRATE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//U2MSGS   DD SYSOUT=*,                                      MIGRATE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
```

Figure 7. CLZJIBM (Part 3 of 4)

```
)IF ACTION=DELETE
//DGLIST   DD SYSOUT=*,                                          DELETE
//         DCB=(LRECL=137,BLKSIZE=3120,RECFM=VBA)
//DGMSGS   DD SYSOUT=*,                                          DELETE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//DGREPT   DD DSN=&&DELLIST,DISP=(MOD,PASS)
//DGEXIT   DD DSN=&&DELEXIT,DISP=(NEW,DELETE),                   DELETE
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=160,BLKSIZE=3200,RECFM=FB)
)ENDIF
)IF ACTION=AUTHCODE
//AUTHMSGS DD SYSOUT=*,                                          AUTHCODE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//AUTHREPT DD SYSOUT=*,                                          AUTHCODE
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FBA)
)ENDIF
)IF ACTION=VERRECOV
//DBUMSGS  DD SYSOUT=*,                                          VERRECOV
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
//*-------------------------------------------------------------------
)IF ACTION=DELETE
//DELMSGS EXEC PGM=IEBGENER
//SYSUT1   DD DSN=&&DELLIST,DISP=(OLD,PASS)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//*
//DELETE  EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=200
//STEPLIB  DD  DSN=CLZ.SCLZLOAD,DISP=SHR
//SYSTSIN  DD  *
  EX 'CLZ.SCLZCGI(CLZTRJDL)'
//SYSTSPRT DD  SYSOUT=*
//DELLIST  DD DSN=&&DELLIST,DISP=(OLD,PASS)
//LISTOUT  DD SYSOUT=*,
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//UNIXLOC  DD DSN=CLZ.SCLZCGI(CLZTULOC),DISP=SHR
)ENDIF
```

Figure 7. CLZJIBM (Part 4 of 4)

Modify CLZJDYN REXX shell

1. Using ISPF EDIT, access member CLZJDYN in the SCLZJCL data set you offloaded from the installation tape.
2. Substitute your site-specific values (identified on the "Cloud 9 installation worksheet" on page 7) as per the instructions in the comment area.
3. Substitute the ISPF data set names for your installation.
4. Save the member.

Figure 8 on page 30 shows the CLZJDYN SCLM REXX shell used for dynamic allocation of ISPF libraries for Web based SCLM functions.

```
/* ------------------------------------------------------------- */
/* NAME:     CLZJDYN                                             */
/* PURPOSE:  CLOUD 9 FOR SCLM                                    */
/*           ISPF ALLOCATIONS FOR SCLM WEB BASED FUNCTIONS.      */
/* ------------------------------------------------------------- */
/*                                                               */
/* REQUIRED MODIFICATION:                                        */
/*  1) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.   */
/*      - ISPFQUAL                                               */
/*                                                               */
/*  NOTE: BREEZE USERS MUST MODIFY THIS MODULE PER EMBEDDED      */
/*        INSTRUCTION. BREEZE DATASET NAMES MAY ALSO NEED TO BE  */
/*        MODIFIED.                                              */
/*                                                               */
/* ------------------------------------------------------------- */
/*  RESIDES IN THE HTTP SERVER AT:                               */
/*  ROOTDIR/CLOUD9/JCL/CLZJDYN                                   */
/* ------------------------------------------------------------- */

    ALLOC FI(ISPTLIB) +
        DSN('%TEMPNAME%' +
        'ISPFQUAL.SISPTENU') SHR

    /* COMMENT THE FOLLOWING LINES IF RUNNING BREEZE */
    ALLOC FI(ISPMLIB) DSN('ISPFQUAL.SISPMENU') SHR
    ALLOC FI(ISPSLIB) DSN('ISPFQUAL.SISPSENU') SHR
    ALLOC FI(ISPPLIB) DSN('ISPFQUAL.SISPPENU') SHR
```

Figure 8. CLZJDYN (Part 1 of 2)

```
    /* ** UNCOMMENT THE FOLLOWING LINES IF RUNNING BREEZE * */
    /* ALLOC FI(SYSPROC) +                                  */
    /*     DSN('%TEMPNAME%' +                               */
    /*                      'BZZ.SBZZCLIB') SHR REUSE       */
    /* ALLOC FI(ISPMLIB) DSN('BZZ.SBZZMENU' +               */
    /*                      'ISPFQUAL.SISPMENU') SHR REUSE  */
    /* ALLOC FI(ISPSLIB) DSN('BZZ.SBZZSENU' +               */
    /*                      'ISPFQUAL.SISPSENU') SHR REUSE  */
    /* ALLOC FI(ISPPLIB) DSN('BZZ.SBZZPENU' +               */
    /*                      'ISPFQUAL.SISPPENU') SHR REUSE  */
    /* ************** END OF BREEZE STATEMENTS ************ */

    /* ** UNCOMMENT THE FOLLOWING IF RUNNING JAVA SUPPORT   */
    /* ALLOC FI(SYSEXEC) DSN('CLZ.SCLZCGI') SHR REUSE       */
    /* ALLOC FI(UNIXLOC) DSN('CLZ.SCLZCGI(CLZTULOC)') SHR REUSE*/
    /* ************** END OF JAVA SUPPORT STATEMENTS ****** */

/* ------------------------------------------------------------ */
/* THE FOLLOWING COMMANDS ALLOCATE TEMPORARY ISPF FILES USED    */
/* BY SCLM DURING PROCESSING.                                   */
/* ------------------------------------------------------------ */
    ALLOC FI(ISPTABL) NEW DELETE DSORG(PO) CYLINDERS,+
        SPACE(1,1) DIR(5) LRECL(80) BLKSIZE(19040) RECFM(F,B)
    ALLOC FI(ISPPROF) NEW DELETE DSORG(PO) CYLINDERS,+
        SPACE(1,1) DIR(5) LRECL(80) BLKSIZE(19040) RECFM(F,B)
    ALLOC FI(ISPLOG) NEW DELETE DSORG(PS) CYLINDERS,+
        SPACE(1,1) LRECL(120) BLKSIZE(2400) RECFM(F,B)
    ALLOC FI(ISPCTL1) NEW DELETE DSORG(PS) CYLINDERS,+
        SPACE(1,1) LRECL(80) BLKSIZE(800) RECFM(F,B)
/* ------------------------------------------------------------ */
/* THE FOLLOWING DATASETS ARE USED BY SPECIFIC TRANSLATORS.     */
/* ------------------------------------------------------------ */
    ALLOC FI(SYSPRINT) NEW DELETE DSORG(PS) CYLINDERS,+
        SPACE(1,1) LRECL(120) BLKSIZE(2400) RECFM(F,B)

/* END OF ALLOCATIONS */
```

Figure 8. CLZJDYN (Part 2 of 2)

Modify CLZJMIG

Note: This task is required only if you have enabled CA-Endevor migration in "Step 4: Set up the CIGINI initialization file" on page 12.

1. Using ISPF EDIT, access member CLZJMIG in the SCLZJCL data set you offloaded from the installation tape.
2. Skip the step of adding your job card values. The jobcard is provided from the job card information in your Cloud 9 profile (see "Step 13: Profile Setup IVP" on page 42).
3. Substitute your site-specific values (identified on the "Cloud 9 installation worksheet" on page 7) as per the instructions in the comment area of the JCL.
4. Substitute the ISPF data set names for your installation.
5. Review the number of level ddnames allowed for conversion.
6. Save the member.

You can view the contents of the CLZJMIG JCL Shell in the SCLZJCL data set.

Step 9: Create and populate additional HFS Cloud 9 directories

In this step, you create Cloud 9 UNIX Root and product directories and populate them with the Cloud 9 product and configuration files you modified in the previous steps. Because many of the members contain case sensitive values, **issue the CAPS OFF command** to ensure that these are not automatically changed to uppercase during editing.

The following is the REXX exec CLZCHMOD that is input to the second step of CLZJUNIX.

```
/* ******************************** REXX ************************** */
/* NAME:  CLZCHMOD                                                  */
/* PURPOSE:                                                         */
/* THIS REXX WILL MODIFY THE SECURITY ATTRIBUTES OF THE UNIX BASED   */
/* COMPONENTS COPIED TO USS IN THE CLZJUNIX JCL JOB STREAM.          */
/* MODIFY THE rootdir VARIABLE AS PER THE WORKSHEET VALUES.          */
/* WARNING: THIS MEMBER CONTAINS CASE SENSITIVE INPUT DATA.          */
/* ******************************** REXX ************************** */
 trace all
 call syscalls 'ON'
 address syscall
 CHMOD 'rootdir/cloud9/jcl/CLZJDYN'  755
 CHMOD 'rootdir/cloud9/jcl/CLZJIBM'  755
 CHMOD 'rootdir/cloud9/jcl/CLZJMIG'  755
 CHMOD 'rootdir/httpd.conf'  755
 CHMOD 'rootdir/httpd.envvars' 755
 CHMOD 'rootdir/cloud9/profiles/user1.jpg' 755
 CHMOD 'rootdir/cloud9/profiles/user1.prf' 755
 CHMOD 'rootdir/cloud9/profiles/user2.jpg' 755
 CHMOD 'rootdir/cloud9/profiles/user2.prf' 755
```

Figure 9. CLZCHMOD

Modify CLZCHMOD

1. Using ISPF EDIT, access member CLZCHMOD in the CLZ.SCLZJCL target library.
2. Issue the CAPS OFF command to ensure that case sensitive values are not changed to uppercase.

   WARNING: UNIX files are case sensitive. Do not change the case on any file names contained in this REXX EXEC.
3. Substitute your site-specific values (identified on the “Cloud 9 installation worksheet” on page 7) as per the instructions in the comment area of the member.
4. Save the member.

Modify and submit CLZJUNIX

1. Using ISPF EDIT, access member CLZJUNIX.
2. Issue the CAPS OFF command to ensure that case sensitive values are not changed to uppercase.

   WARNING: UNIX files are case sensitive. Do not change the case on any file names contained in this JCL.
3. Copy your job card values to the top of the member.
4. Substitute your site-specific values (identified on the "Cloud 9 installation worksheet" on page 7) as per the instructions in the comment area of the JCL.
5. Submit the job.

Note: This job should end with COND CODE=0. If it does not:

1. Review your job card parameters and the JCL for errors.
2. Resubmit the job.

```
//*(JOBCARD)
//*  ------------------------------------------------------------------
//*  NAME:    CLZJUNIX
//*  PURPOSE: JCL TO CREATE AND POPULATE THE CLOUD 9 UNIX DIRECTORIES.
//*  USAGE:   Make sure profile of 'caps off' prior to modifying this
//*           member. Unix directory and file names are case sensitive.
//*  USAGE:   Set to 'number off' prior to modifying this
//*           member.
//*  ------------------------------------------------------------------
//*         * * *   N O T I C E   * * *
//*    THIS PROGRAM IS A PROPRIETARY PRODUCT OF CHICAGO INTERFACE
//*    GROUP, INC. @ COPYRIGHT 2003 CHICAGO INTERFACE GROUP, INC.
//*    ALL RIGHTS RESERVED.
//*  ------------------------------------------------------------------
//*
//*  **                                  **
//*  ** PRODUCT INSTALLATION/SETUP ISSUES **
//*  **                                  **
//*  THE FOLLOWING IS A LIST OF MODIFICATIONS REQUIRED DURING PRODUCT
//*  INSTALLATION AND INITIAL SETUP:
//*
//* 1) ADD A VALID JOB CARD
//* 2) CHANGE rootdir to the root directory value in the
//*    in your worksheet.
//* 3) Change USER1 and USER2 to actual userids. Use Upper Case.
//*    These files are demo profile files for the IVP.
//* 4) DO NOT change the case on the file names. Unix files are
//*    case sensitive.
//* 5) Ensure that the last step points to the dataset that contains
//*    the REXX member CLZCHMOD.
//*
//CMD0     EXEC PGM=IKJEFT01,REGION=4096K,TIME=1439,DYNAMNBR=200
//*-------------------------------------------------------------------
//* TSO OUTPUT FILE
//*-------------------------------------------------------------------
//SYSTSPRT DD SYSOUT=(*)
//*-------------------------------------------------------------------
//* TSO INPUT FILE
//*-------------------------------------------------------------------
//SYSTSIN  DD *
```

Figure 10. CLZJUNIX (Part 1 of 2)

```
 MKDIR 'rootdir/cloud9/jcl'  MODE(7,5,5)
 MKDIR 'rootdir/cloud9/profiles'  MODE(7,7,7)
 MKDIR 'rootdir/cloud9/manifest'  MODE(7,7,5)
 MKDIR 'rootdir/cloud9/cache'     MODE(7,7,5)
 MKDIR 'rootdir/logs'  MODE(7,7,7)
 MKDIR  'rootdir/reports'  MODE(7,7,7)
 OPUT 'CLZ.SCLZJCL(CLZJDYN)' -
       'rootdir/cloud9/jcl/CLZJDYN'
 OPUT 'CLZ.SCLZJCL(CLZJIBM)' -
       'rootdir/cloud9/jcl/CLZJIBM'
 OPUT 'CLZ.SCLZJCL(CLZJMIG)' -
       'rootdir/cloud9/jcl/CLZJMIG'
 OPUT 'CLZ.SCLZHTML(CLZHTTPD)' -
       'rootdir/httpd.conf'
 OPUT 'CLZ.SCLZHTML(CLZEVARS)' -
       'rootdir/httpd.envvars'
 OPUT 'CLZ.SCLZJPG(CLZCIG01)' -
       'rootdir/cloud9/profiles/user1.jpg' BINARY
 OPUT 'CLZ.SCLZJPG(CLZCIG02)' -
       'rootdir/cloud9/profiles/user2.jpg' BINARY
 OPUT 'CLZ.SCLZPRF(CLZCIG01)' -
       'rootdir/cloud9/profiles/user1.prf'
 OPUT 'CLZ.SCLZPRF(CLZCIG02)' -
       'rootdir/cloud9/profiles/user2.prf'
/*
//CHMOD0   EXEC PGM=IRXJCL,PARM=(CLZCHMOD)
//*-------------------------------------------------------------------
//* REXX STANDARD FILES
//*-------------------------------------------------------------------
//SYSTSPRT DD SYSOUT=(*)
//SYSTSIN  DD DUMMY
//*-------------------------------------------------------------------
//* THE FOLLOWING DATASET MUST CONTAIN THE REXX MEMBER CLZCHMOD
//*-------------------------------------------------------------------
//SYSEXEC  DD  DSN
```

Figure 10. CLZJUNIX (Part 2 of 2)

Note: After this job has been submitted and executes successfully, the Cloud 9 USS directories should be populated and ready for testing.

Step 10: Review authorization requirements for CLZRSDRV

The module CLZRSDRV is the interface module for invoking authorized, real-time processes in the HTTP server.

To check the attributes of the CLZRSDRV authorized program interface module, use one of the following methods:

- ISPF UNIX shell (requires that the SYS1.SBPXxxxx libraries are in your logon setup)
- OMVS Command shell

Using the ISPF UNIX shell

This option requires that the SYS1.SBPXxxxx libraries are in your logon setup.

1. Access UNIX System services

   ```
   tso %ishell
   ```

2. Display the rootdir/cgi-bin directory. When the command line appears, type

   ```
   rootdir/cgi-bin/
   ```

where *rootdir* is the name of the root directory used by your installation.

3. Issue the attribute 'a' line command for CLZRSDRV.

```
                                    Directory List
                                                          Command===>
_____________________________________________________________________

/u/ibmdemo/cgi-bin/
 Select one or more files with / or action codes.

   Type  Filename                                              Row 1 of 19
_ Dir   .
_ Dir   ..
_ File  CLZRADDS
_ File  CLZREDRV
_ File  CLZRENDV
_ File  CLZRINDX
_ File  CLZRLMBR
_ File  CLZRLUNX
_ File  CLZRMENU
_ File  CLZRMLST
_ File  CLZRPROF
_ File  CLZRSCLM
_ File  CLZRSCMA
a File  CLZRSDRV
_ File  CLZRSDSF
```

Figure 11. Display of rootdir/cgi-bin Directory

Note: The number of actual members in the CGI-BIN directory is subject to change.

4. From the Edit pull down menu, select Option 1 Mode fields.

```
  .-------------------------------------------------.
/ |  Edit  Help                                     |
S |  .-----------------------------. ---------------  |
  |  |    1. Mode fields...        | es              |
  |  |    2. Owning user...        |                 |
_ |  |    3. Owning group...       |                 |
_ |  |    4. User auditing...      |     More:     + |
_ |  |    5. Auditor auditing...   |                 |
_ |  |    6. File format...        |                 |
_ |  |    7. Extended Attributes... |                 |
_ |  '-----------------------------'                 |
a |  Group owner  . . . : SYS1(0)                    |
_ |  Last modified  . . : 10/11/2000 21:59 GMT       |
_ |  Last changed . . . : 10/11/2000 21:59 GMT       |
_ |  Last accessed  . . : 10/11/2000 20:26 GMT       |
  |  Created  . . . . . : 10/11/2000 20:26 GMT       |
_ |  Link count . . . . : 1                          |
_ |  Set UID bit  . . . : 0                          |
_ |                                                  |
_ |                                                  |
_ '--------------------------------------------------'
```

Figure 12. Edit Pull Down — Mode Fields

5. In the "Change the Mode" panel, ensure that the sticky bit (the access permission setting) is set to 1.

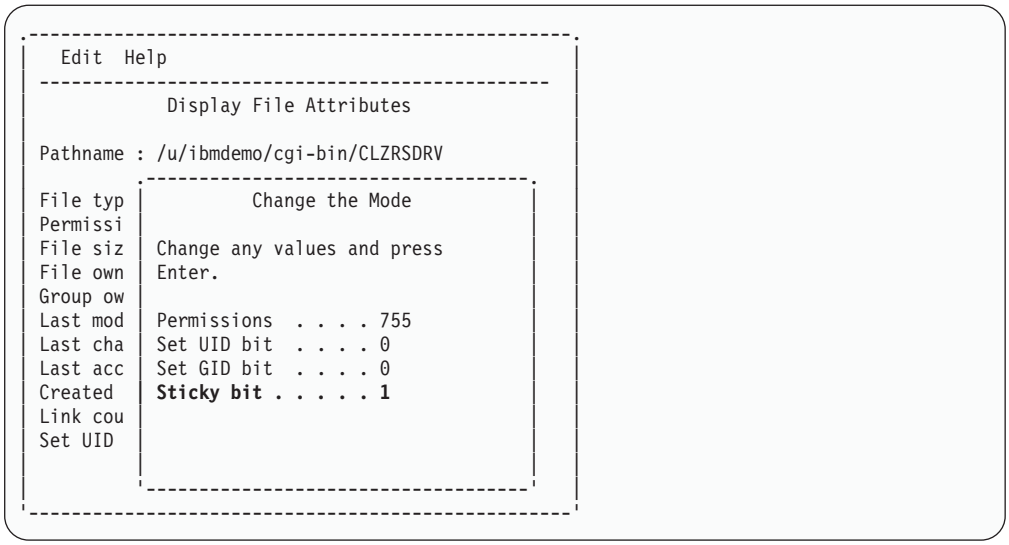

Figure 13. Change the Mode Panel

Using the OMVS command shell

1. Access UNIX System services using one of the following commands:
   - From TSO Ready:

     `OMVS`
   - From an ISPF panel:

     `TSO OMVS`
   - Telnet to UNIX System services: contact your system administrator to find if this function has been enabled on your system and what address:port to use.
2. Change to the rootdir/cgi-bin directory:

   `cd rootdir/cgi-bin`
3. Display the attributes for CLZRSDRV:

   `ls -l CLZRSDRV`

   The first column of output is the attribute bits, which should be:

   `-rwxr-xr-t`

Trouble shooting

If you encounter any problems with this step, double-check that the following items are in place:

- The real module CLZRSDRV and its required alias C9RSDRV reside in the linklist or steplib.
- A dummy stub entry for CLZRSDRV, with a length of zero, resides in the cgi-bin directory.
- The CLZRSDRV file has the access permission enabled (referred to as ″sticky bit″ in USS terminology).

CA-Endevor Bridge

If you are using the CA-Endevor Bridge CIGINI option, you also need to perform “Step 10: Review authorization requirements for CLZRSDRV” on page 34 for the rootdir/cgi-bin file called CLZREDRV.

Checkpoint #3 for Cloud 9 base product installation

At this point the host based modification and configuration work should be complete. Before continuing with the next part of the installation, which involves copying files to UNIX and performing IVPs, verify that the steps listed in the following table have been completed.

Table 12. Checkpoint #3 for Cloud 9 base product installation

Task	Completed?
CLZHTTPD has been modified?	
CLZEVARS has been modified?	
CLZC9SRV has been reviewed and modified?	
The user ID and password on the server JCL has the authority to submit a server task?	
The server jobname is the same as the WEBJOBNAME parameter in the CLZHTTD member?	
The CLZJIBM batch JCL member has been reviewed and modified?	
The CLZJDYN allocation shell has been reviewed and modified?	
The CLZJMIG batch JCL member has been reviewed and modified?	
The steplib in the CLZC9SRV JCL is the same as in CLZJIBM and CLZJMIG?	
Is the steplib in CLZC9SRV JCL authorized?	

Configure USS and HTTP Server components - Cloud 9 installation

Chapter 5. Perform Installation Verification Procedures

Step 11: HTTP Server invocation IVP

To test that the HTTP Server can be correctly invoked for Cloud 9, you need to:

- Start the server
- Shut down the server
- Restart the server (in preparation for the next IVP)

Start the server

1. Submit the CLZC9SRV job, located in the SCLZJCL data set.
2. View the //SYSPRINT DD and //SYSOUT DD in the job output and verify that it looks like the output in Figure 14.

```
******************************** TOP OF DATA *******
IMW0234I Starting.. httpd
IMW0235I Server is ready.
******************************** BOTTOM OF DATA *****
******************************** TOP OF DATA **********************
..... This is IBM HTTP Server V5R1M0
..... Built on Feb 17 1999 at 20:10:29.
..... Started at Sat Mar 25 16:17:31 2000
..... Running as "P390", UID:0, GID:0.
******************************** BOTTOM OF DATA ******************
```

Figure 14. SYSPRINT DD AND SYSOUT DD

Shut down the server

To quiesce the server job, enter one of the following commands (replace *cloud9–job-name* with the name of your job):

If entered on an MVS console:

`STOP cloud9-job-name`

If entered in a console interface, such as SDSF:

`/STOP cloud9-job-name`

Note: Because the HTTP Server uses TCP/IP stack and a cancel does not always clean up storage, quiesce the server by using the MVS console command method, rather than by canceling the job. When you issue the console command, the Cloud 9 HTTP Server job ends cleanly.

Restart the server

1. Re-submit the server JCL (CLZC9SRV) for the next test.

Checkpoint #4 for Cloud 9 base product installation

At this point, you should have successfully completed the following tasks:

Table 13. Checkpoint #4 for Cloud 9 base product installation

Task	Completed?
Submitted the server JCL — CLZC9SRV?	
Reviewed the sysout files showing the port # and verifying that this is port # you expected?	
Issued a Quiesce of the server to test command and clean up?	
Resubmitted the server for the next test?	

Step 12: Cloud 9 Invocation and Logon IVP

This test verifies that the base product has been correctly configured and that Cloud 9 is accessable through the Web. The most probable causes of failure in this step are incorrect security settings or incorrect configuration.

Execute cloud9.htm

To test installation of the application, execute the cloud9.htm file directly from HTTP server directories, as follows:

1. On your desktop, start your browser.
2. Modify the following statement with your IP address and port number and type it in the browser's address window (labeled "Location" in Netscape Navigator and "Address" in Internet Explorer):

   `http://ip-address:portno/cloud9.htm`

   The browser requests the html file directly from HTTP and executes the Cloud 9 application.
3. When the Cloud 9 product is invoked, you are prompted with a log-in window. Enter your TSO user ID and password and click "ok" to begin using Cloud 9.

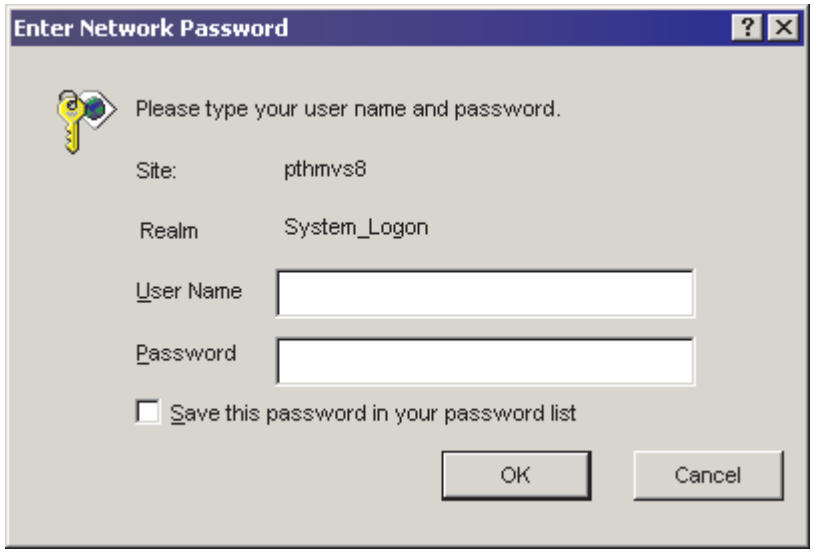

Figure 15. Cloud 9 Logon Prompt

Diagnostic checks

If you cannot invoke the Cloud 9 application, use the following list to attempt to determine the problem:

1. Is the Cloud 9 server active? How do you know?
2. Are there any error messages in the SYSOUT queue or on the CONSOLE?
3. From your browser are you using the correct *ip-address:port* combination? How do you know?
4. Are you using a Cloud 9 supported browser (Netscape 4.7 to 6.x, or Explorer 5.0 or higher)?
5. Can you access any other HTTP server applications?

Verify that the UNIX directories are configured as outlined in Appendix A, "Cloud 9 UNIX directory structure," on page 207 in this manual.

Step 13: Profile Setup IVP

During the execution of CLZJUNIX, a profile and picture for two user IDs was stored in the Cloud 9 root directory. If you are logged on as one of those user IDs, then you can view the profile at this time.

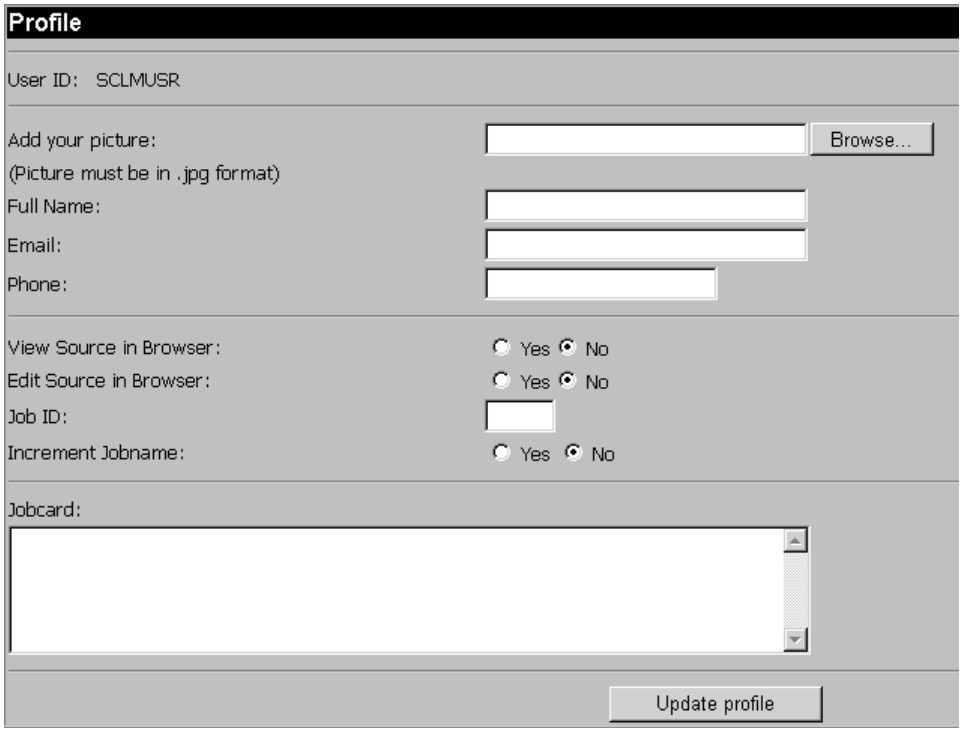

Figure 16. Profile page

Select **PROFILE** on the Main Menu to set up your profile at this time with real values and JOB CARD information.

Tip: If this step fails with a message similar to "No data received from host" (the exact wording of the message depends on the Web browser being used), the most likely cause is that the TCPIP.SEZATCP library is inaccessible. In this case, examination of the SYSLOG shows an 806 ABEND for module EZACICnn (where nn depends on your local configuration). Add TCPIP.SEZATCP to the STEPLIB for the Cloud 9 server and restart it.

Note: This library (and any other library in the STEPLIB concatenation) must be APF authorized. This should have been tested in "Step 5: Run environment diagnostic tests" on page 16.

Step 14: Batch and Interactive IVPs

To test the connection to the host, perform the following steps:

1. Select **List SCLM Files** from the Main Menu.
2. On the SCLM Query page, fill in a valid SCLM project name and optionally other filters.
3. Click **Submit**.

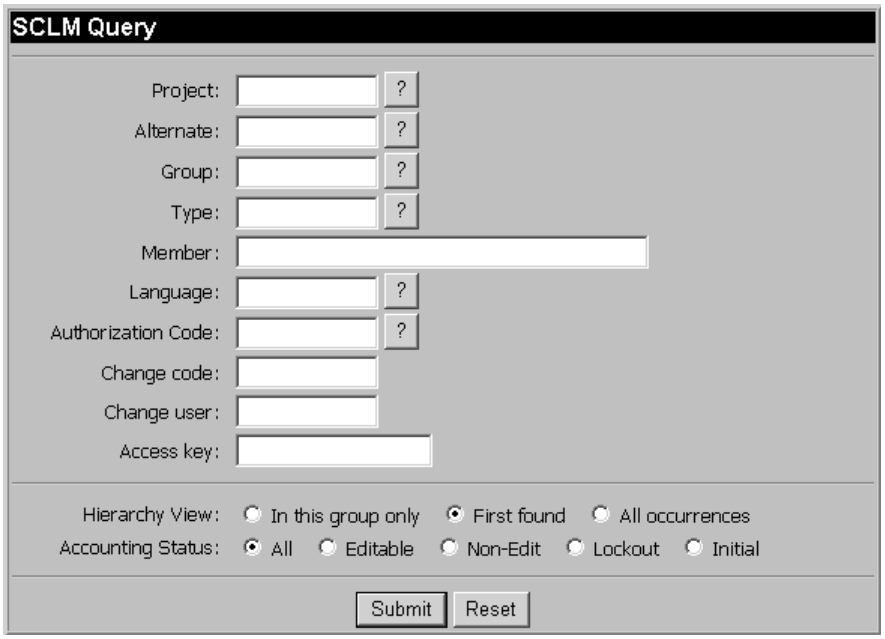

Figure 17. SCLM Query page

4. Verify that the list returned has the same format as that shown in Figure 18:

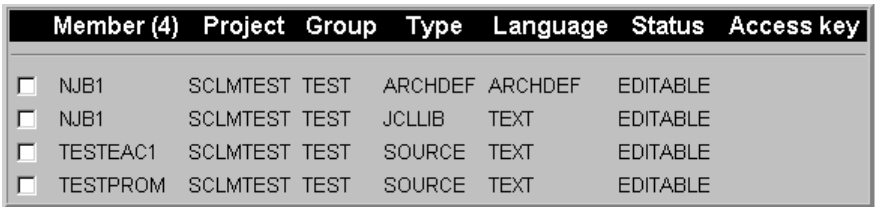

Figure 18. Search List page

5. Use this list to perform batch and interactive IVP processes.

Tip: If this step fails (the panel returned has the messages *Failure to call TSO/ISPF/SCLM* and *NULL FILE: SYSTSPRT*), it is most likely because one or more STEPLIB data sets are lacking APF authorization. Verify that authorization is properly specified in the PROG00 member of SYS1.PARMLIB.

Test the Batch Interface:

- Check one of the members on the list
- Select **Build** action.
- Fill in all options (including selection of batch submission) and click **Submit**.
- Check the expansion of the JCL in the JES2 Hold queue to validate that CLZJIBM was modified correctly.
- Review the batch JCL that was submitted and ensure CLZJIBM was found and modified correctly.

Exit Cloud 9:

To close your browser, either:

- Select **Close** from the file pull-down menu, Or

- Click the **X** button in the upper right hand corner of the browser window.

Step 15: Perform Batch SLR IVP

Modify and Submit CLZC9J06

Earlier in the installation, you created a demo version of the SLR database. At this time, execute the SLR IVP to review the CIGINI setup and the demo database display and update. For information about SLR syntax, see Appendix B, "Suite Long Name Registry," on page 213.

1. Using ISPF EDIT, access member CLZC9J06 in the SCLZJCL data set.
2. Add a valid job card.
3. Substitute your site-specific values (identified on the "Cloud 9 installation worksheet" on page 7) as per the instructions in the comment area of the JCL.
4. Submit the member.
5. Review output.

Note: Before any work is carried out on members in libraries that will have long file name support, ensure that you have defined the NAME RULE for them in the SLR. Defining the rule after members have been created can cause inconsistant results when those members are subsequently modified and saved.

Figure 19 on page 45 shows the CLZC9J06 member used to test the SLR setup and update.

```
//**(JOBCARD)
//**
//*********************************************************************
//* CLOUD 9 FOR SCLM VERSION OF IVP                                   *
//*********************************************************************
//*                                                                   *
//* CLZC9J06 - THE PURPOSE OF THIS JCL TO RUN THE SLR DATA IVP.       *
//*            STEP 1 WILL PRINT THE CIGINI DEFINITIONS.              *
//*            STEP 2 WILL LIST IVP SLR RULE DEFINITIONS.             *
//*            STEP 3 WILL ADD DATASET AND SCLM TYPE DEFINITIONS      *
//*                   AND THEN LIST ALL RULES IN THE DATABASE.        *
//* NOTE:    - THE SYNTAX PROVIDED IS FOR AN EXAMPLE ONLY.            *
//*            IT IS RECOMMENDED THAT STEP3 SYNTAX BE TAILORED TO     *
//*            ACTUAL LOCAL VALUES.                                   *
//*********************************************************************
//*                                                                   *
//* REQUIRED JCL MODIFICATION:                                        *
//*   1) INCLUDE A JOBCARD                                            *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//*  STEP 1: PRINT THE CIGINI DEFINITIONS.                            *
//*                                                                   *
//*********************************************************************
//STEP1    EXEC PGM=CLZNTINI
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR
//CIGPRINT DD SYSOUT=*
//*********************************************************************
//*                                                                   *
//*  STEP 2: LIST THE CURRENT CONTENTS OF THE SLR DATABASE            *
//*                                                                   *
//*********************************************************************
//STEP2    EXEC PGM=CLZSLR
//STEPLIB DD DSN=CLZ.SCLZLOAD,DISP=SHR
//CIGPUNCH DD SYSOUT=*
//CIGLOG   DD SYSOUT=*
//CIGIN   DD *
 LIST NAME RULES.
/*
//*********************************************************************
//*                                                                   *
//*  STEP 3: ADD DATASET AND TYPE DEFINITIONS TO SLR DATABASE.        *
//*          USE AS IS OR TAILOR WITH LOCAL VALUES.                   *
//*                                                                   *
//*********************************************************************
//STEP3    EXEC PGM=CLZSLR
//STEPLIB DD DSN=CLZ.SCLZLOAD,DISP=SHR
//CIGPUNCH DD SYSOUT=*
//CIGLOG   DD SYSOUT=*
//CIGIN   DD *
 ADD NAME RULE FOR DATASET 'CLZ.SCLZLOAD.PDS1' CASE SENSITIVE.
 ADD NAME RULE FOR DATASET 'CLZ.SCLZLOAD.PDS2' CASE INSENSITIVE.
 ADD NAME RULE FOR SCLM  TYPE HTML     .
 ADD NAME RULE FOR SCLM TYPE JAVA      CASE SENSITIVE .
 ADD NAME RULE FOR SCLM TYPE JAVACLAS  CASE SENSITIVE .
 ADD NAME RULE FOR SCLM TYPE JAR       CASE SENSITIVE .
 ADD NAME RULE FOR SCLM TYPE DOC       CASE INSENSITIVE .
 LIST NAME RULES.
/*
```

Figure 19. CLZC9J06

Step 16: Setup SLR Maintenance JCL

Modify CLZC9J04

At this time the basic installation is complete. There are a number of other jobs that can be tailored to perform a number of functions against the SLR database. These are described as follows:

CLZC9J03
This JCL can be used to create and initialize an empty SLR database. The database you have already created for installation verification testing has demo data contained within it. If you want to set up a new database from scratch, then use this job.

CLZC9J04
This JCL can be used to set up a backup, delete and define a JCL stream for production use.

CLZC9J05
This JCL can be used to expand the vsam indexes for the SLR. This step is also included in many of the other SLR JCL streams.

For all of these jobs, access the relevent JCL member in the SCLZJCL data set and tailor as follows:

1. Add a valid job card.
2. Substitute your site-specific values (identified on the "Cloud 9 Installation Worksheet" on page 9) as per the instructions in the comment area of the member.
3. Save the member.

Figure 20 shows the CLZC9J04 member that contains the SLR file maintenance JCL.

```
//**(JOBCARD)
//**
//*********************************************************************
//*                                                                   *
//* CLZC9J04 - THE PURPOSE OF THIS JCL IS TO BACKUP, DELETE, AND      *
//*            DEFINE A PRODUCTION SLR DATABASE.                      *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//* REQUIRED JCL MODIFICATION:                                        *
//*   1) INCLUDE A JOBCARD                                            *
//*   2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.      *
//*      - VOLUMES(DVOLSER)                                           *
//*      - DUNIT                                                      *
//*      - TDISK                                                      *
//*   3) SIZE THE FILES IN STEP2, STEP3, AND STEP4.                   *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//* DO NOT MODIFY THE VSAM PARAMETERS PROVIDED IN THIS JCL. DOING SO *
//* WILL PRODUCE UNEXPECTED RESULTS FROM THE CLOUD9 APPLICATION.      *
//*                                                                   *
//*********************************************************************
```

Figure 20. CLZC9J04 (Part 1 of 3)

```
//*                                                                   *
//* STEP1: DELETE THE OLD VERSION OF THE BACKUP, IF IT EXISTS.        *
//*        DELETE THE OLD VERSION OF THE SORT FILE. IF IT EXISTS.     *
//* STEP2: CREATE A SEQUENTIAL BACKUP OF THE SLR DATABASE USING       *
//*        STANDARD IDCAMS REPRO SERVICES.                            *
//* STEP3: SORT THE DATA.                                             *
//* STEP4: DELETE, DEFINE, AND REPRO THE SLR DATABASE.                *
//* STEP5: EXPAND VSAM INDEXES ON THE SLR DATABASE.                   *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//* STEP1: DELETE THE OLD VERSION OF THE BACKUP, IF IT EXISTS.        *
//*                                                                   *
//*********************************************************************
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
      DELETE 'CLZ.SCLZSLR.SLR.SEQ'       PURGE
      DELETE 'CLZ.SCLZSLR.SLR.SORT' PURGE
      IF MAXCC <= 8 THEN DO
        SET MAXCC = 0
        SET LASTCC = 0
      END
//*********************************************************************
//*                                                                   *
//* STEP2: CREATE A SEQUENTIAL BACKUP OF THE SLR DATABASE USING       *
//*        STANDARD IDCAMS REPRO SERVICES.                            *
//*                                                                   *
//*********************************************************************
//STEP2    EXEC PGM=IDCAMS,
//         COND=(0,LT)
//OUTDD02  DD DSN=CLZ.SCLZSLR.SLR.SEQ,DISP=(NEW,CATLG,DELETE),
//         UNIT=DUNIT,SPACE=(CYL,(10,5),RLSE),
//         DCB=(RECFM=VB,LRECL=604,BLKSIZE=6160)
//INDD02   DD DSN=CLZ.SCLZSLR.DATABASE.DATA,DISP=OLD,
//            AMP='BUFNI=10,BUFND=10'
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  REPRO IFILE(INDD02)  OFILE(OUTDD02)
/*
//*********************************************************************
//*                                                                   *
//* STEP3:  SORT THE DATA AND DELETE ALL RECORDS THE QUALIFY FOR A    *
//*         LOGICAL DELETE.                                           *
//*                                                                   *
//*********************************************************************
```

Figure 20. CLZC9J04 (Part 2 of 3)

```
//STEP3    EXEC PGM=SORT,
//         COND=(0,LT)
//SORTIN   DD DSN=CLZ.SCLZSLR.SLR.SEQ,DISP=SHR
//SORTOUT  DD DSN=CLZ.SCLZSLR.SLR.SORT,DISP=(NEW,CATLG,DELETE),
//         UNIT=DUNIT,SPACE=(CYL,(10,5),RLSE),
//         DCB=(RECFM=VB,LRECL=604,BLKSIZE=6160)
//SORTWK01 DD   UNIT=TDISK,SPACE=(CYL,(5,5))
//SORTWK02 DD   UNIT=TDISK,SPACE=(CYL,(5,5))
//SORTWK03 DD   UNIT=TDISK,SPACE=(CYL,(5,5))
//SORTWK04 DD   UNIT=TDISK,SPACE=(CYL,(5,5))
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
  SORT FIELDS=(5,254,CH,A)
  RECORD TYPE=V,LENGTH=(604,,,254)
  SUM FIELDS=NONE
/*
//*********************************************************************
//*                                                                   *
//*  STEP 4: ALLOCATE THE SLR VSAM DATABASE AND REPRO BACKUP          *
//*                                                                   *
//*********************************************************************
//STEP4   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INDD01   DD DSN=CLZ.SCLZSLR.SLR.SORT,DISP=SHR
//SYSIN   DD *
 DELETE CLZ.SCLZSLR.DATABASE
 DEFINE CLUSTER -
    (NAME('CLZ.SCLZSLR.DATABASE') -
    IMBED SPEED UNIQUE FREESPACE(30 30) -
    VOLUMES(DVOLSER) TRACKS(60 40) -
    SHR(4 3) -
    KEYS(254 0) -
    RECORDSIZE(512 1024)) -
  DATA  (CISZ(16000)) -
  INDEX (CISZ(4096))
 REPRO INFILE(INDD01) OUTDATASET('CLZ.SCLZSLR.DATABASE')
/*
//*********************************************************************
//*                                                                   *
//*  STEP 5: FORCE A VSAM SPLIT FOR INTEGRITY SUPPORT.                *
//*                                                                   *
//*********************************************************************
//STEP5    EXEC PGM=CLZVSM2L,PARM='CLZ.SCLZSLR.DATABASE'
//STEPLIB DD DSN=CLZ.SCLZLOAD,DISP=SHR
```

Figure 20. CLZC9J04 (Part 3 of 3)

CHECKPOINT #5 for Cloud 9 Installation

At this point, you should have successfully completed the following tasks:

Table 14. Checkpoint #5

Task	Completed?
Invoked the Cloud 9 application?	
Logged onto the application, passing the security check?	
Viewed demo profile and updated with real data?	
Displayed members and ran SCLM jobs?	
Exited successfully from Cloud 9?	
Set up the backup, delete, and define JCL for the SLR?	

Part 2. SCLM-Java Development Kit for USS Build and Deploy

Chapter 6. S-JDK for USS build and deploy installation overview

This part of the manual contains the installation procedure for the IBM Cloud 9 for SCLM for z/OS SCLM-Java Development Kit feature for the UNIX Systems Services. Hereinafter, the following names are used in this manual:

- IBM Cloud 9 for SCLM for z/OS is called ***Cloud 9***
- IBM Cloud 9 for SCLM for z/OS Java Development Kit is called ***S-JDK.***
- UNIX System Services and HFS are called ***USS***.

The steps in this part are organized into four major sections:

- Before you begin
- Customizing translators and translator control files
- Defining the S-JDK inventory, USS and Cloud 9 parts
- Performing Installation Verification Procedures (IVP)

READ THIS FIRST!

This is not an 'out of the box' solution. You should not use global editing on these files. You must thoroughly understand your JAVA/USS environment before attaching the SCLM translators.

The SCLM part of this solution involves standard SCLM administrator tasks. There are translators, types, languages and control files to be reviewed and modified. However, because some SCLM administrators might be unfamiliar with the JAVA/USS side of the setup, you need to thoroughly review the system and software requirements for S-JDK (see Chapter 7, "Before you begin," on page 57), **paying particular attention to verifying the Java/USS environment ("Verify the Java/USS environment" on page 57)**, before moving on to setting up the prototype translators.

Overview of the S-JDK for USS

The S-JDK for USS provides the developer with the means to add to SCLM e-business type objects, such as wordprocessing documents, spreadsheets, graphics files, HTML, XML and Java objects. It also provides a means with which to compile the Java programs to create class files and use the Java Jar process to put all the required artifacts into a Java archive (Jar). In this part of the guide, you set up the translators and control files that control the e-business object management using z/OS UNIX Systems Services (USS). In part 3, the translators and control files to set up the same process but using a remote server, such as an NT server, are described.

The three translators provided with this release, CLZTJAVA, CLZTJAR and CLZTJBIZ, are used to manage and build Java Source, Jar make files and other binary and text e-business objects. These are standard SCLM translators that use control files to drive the copying and compiling of the e-business objects.

It should be noted that, at all times, the SCLM PDS's are the repository for all the code. The USS is used as an area in which to build, store and run e-buiness type applications such as HTML and Java. Source is copied there by the build process

and outputs, such as Java class files, are copied back into SCLM by Cloud 9, on successful completion of a compile. This ensures that class files for a particular Java source file are all kept together in SCLM. In this section, you tailor the mapping control files to show the SCLM life cycle name and the USS directory to which it maps. In this way, Cloud 9 can copy to and from the USS directories during the build processes.

You can use the Cloud 9 SLR database (Short to Long name Registry) to store objects that have long file names, such as a Windows or UNIX file named ProjectOverview.doc, in SCLM. When you add a UNIX or PC file that has a long name to SCLM, Cloud 9 stores the long name in the SLR, along with a short name that it generates. This makes it possible to store members that conform to MVS naming conventions in SCLM PDS's, but maintain a record of their actual long file name in the SLR. Anytime you work with these members in the Cloud 9 lists, you see the long name but if you look at them from native SCLM, you see the generated short name.

The following steps give an overview of what happens during a Java program compile. To see how each of these steps is performed, check the User Guide. This overview is used to inform you how the translators and control members fit into the process. You perform these steps as part of the IVP after tailoring has occured.

1. Migrate the Java Source into SCLM, giving it a language of JAVA. The language is related to the LANG= parameter in the required translator (CLZTJAVA).
2. Issue a Build against the member in Cloud 9. At this time Cloud 9 invokes the CLZTJAVA translator.
3. The CLZTJAVA translator initially goes to the SLR to get the short name, so that it knows what the actual SCLM member is called.
4. It then uses the CLZTULOC to see the USS directory where it is going to copy the Java source, based on its actual location within SCLM. When it does the copy, it copies from SCLM to USS using the short name in SCLM and the long name in USS.
5. The translator then builds the Java compile shell from the CLZTJAVC control member, using the CLZTCPTH control member to define the class locations. This is used to tell Java where, within the SCLM hierarchy, included class files can be found.
6. A Command file, generated from these two control members, is created and copied to the USS directory that contains the Java Source being compiled. From here, the Command file is executed. When executed, Java class files and a listing are placed in the USS locations specified in the CLZTULOC control member.
7. The translator then creates short names for them in the SLR (if the names don't already exist), and copies them back into the SCLM groups and types specified in CLZTULOC.

Modifying case-sensitive S-JDK files

During this installation, you modify several JCL members and USS files. Certain JCL members and all USS files contain case-sensitive values. It is imperative that *before* modifying the JCL and USS members, you issue the CAPS OFF command to ensure that automatic upper casing does not occur. You are reminded of this case sensitivity issue where appropriate throughout this manual.

Translators and translator control files

S-JDK translators

CLZTJBIZ Translator for e-business types, such as HTML, XML, Graphics (JPEG, GIF), and other types.

CLZTJAVA Translator for JAVA

CLZTJAR Translator for JAR

S-JDK USS translator control files

CLZTJARU

- Input to USS CLZTJAR Translator
- Compile shell for USS JAR type

CLZTJMAP

- Input to USS CLZTJAR Translator
- USS Output Control for USS JAR type

CLZTJAVC

- Input to USS CLZTJAVA Translator
- Compile Shell for USS JAVA type

CLZTCPTH

- Input to USS JAVA and JAR Translators
- %classpath% Substitution.

CLZTULOC

- Input to all USS S-JDK Translators
- SCLM to USS Directory mapping rules.

CLZTHTPD

- Input to USS CLZTJAVA Translator
- ADDTYPE list for Java compiles

A step-by-step approach

Table 15. S-JDK for USS installation steps

Before you begin...	
1.	Review system and software considerations.
2.	Determine Inventory Values and Type Definitions
CP1.	Verify steps as shown in “Checkpoint #1 for S-JDK for USS installation” on page 61
Customize translators and translator control files	
3.	Review and modify all translators and translator control file members.
CP2.	Verify steps as shown in “Checkpoint #2 for S-JDK for USS installation” on page 82.
Define the S-JDK inventory, USS and Cloud 9 parts	
4.	Modify and run CLZTALIB, CLZTAVSM, and CLZTPDEF to build S-JDK Project Definitions
5.	Modify and run CLZC9J06 to define S-JDK types to Cloud 9
6.	Modify and run CLZTAUNX to define USS directories
7.	Review CLZJIBM UNIX Shell
CP3.	Verify steps as shown in “Checkpoint #3 for S-JDK for USS Installation” on page 100.
Perform Installation Verification Procedures (IVP)	
8.	Add Clock2.java and Clockh.html IVP programs
9.	Invoke Clockh.html to display Time of Day Java IVP
CP4.	Verify steps as shown in “Checkpoint #4 for S-JDK for USS installation” on page 103.

Chapter 7. Before you begin

Step 1: Review software and assumptions

In this step, you review the system and software requirements for S-JDK for USS installation.

System requirements

To successfully install Cloud 9 S-JDK for USS, the following system requirements must be in place at your installation:

Table 16. System requirements

z/OS Operating System	Version 2 Release 7 (or higher)
SCLM	Standard z/OS
Java/USS	Enabled USS environment
IBM Cloud 9	Cloud 9 installed and configured

Verify the Java/USS environment

- Verify that you have access to USS environment. Check with your USS Administrator for information about your USS environment.
- Verify which USS directories contain the z/OS Java compiler and class files.
- Obtain sample JCL to run a compile standalone in batch to verify that you have access and security rights in this environment.

Check other documents that can be useful

There are several Redbooks available from IBM concerning USS and JAVA.

- *Debugging UNIX System Services*
- *e-business Enablement Cookbook for OS/390 Volumes 1,2,3*

For the publication order numbers for these books, see "Where to find more information" on page xi.

For information regarding USS setup, the *UNIX System Services Planning* manual (GA22-7800-01) might be useful, in particular, Chapter 14.

Assumptions

Java/USS Environment is already configured.

Users who are Building and Promoting Java components have defined an OMVS segment that includes a UID and home directory to RACF (or other security product).

Step 2: Determine inventory values and type definitions

Determine SCLM and USS inventory values

The S-JDK is set up with default values. These default values are meant to be modified throughout the various translators, HTML and JCL members. Before making modifications to these members, review the following worksheets and fill in your site specific inventory values.

It is important to view the SCLM Inventory and the USS directory structure as extensions to each other. Review both tables before making decisions about the values.

SCLM inventory value worksheet

Table 17. SCLM inventory value worksheet

SCLM Inventory Name	Default Value	Your Values
Project	IBMDEMO	
Alt-project	IBMDEMO	
Group	DEV,QA,REL	
Types	JAVA,JAR,HTML,GRAPHICS	
Languages	JAVA,JAR,EBIZ	

USS directory value worksheet

Table 18. USS Directory Value Worksheet

USS Mapping Locations	Default Value	Your Values
Listings	• /u/ibmdemo/dev/listings • /u/ibmdemo/qa/listings • /u/ibmdemo/rel/listings	
Classes	• /u/ibmdemo/dev/classes • /u/ibmdemo/qa/classes • /u/ibmdemo/rel/classes	
Graphics	• /u/ibmdemo/dev/graphics • /u/ibmdemo/qa/graphics • /u/ibmdemo/rel/graphics	
Html	• /u/ibmdemo/dev/html • /u/ibmdemo/qa/html • /u/ibmdemo/rel/html	
Jar	• /u/ibmdemo/dev/jar • /u/ibmdemo/qa/jar • /u/ibmdemo/rel/jar	

Review SCLM and Cloud 9 type definitions

This step documents which Types need to be defined to SCLM and Cloud 9. Before moving on to the Translator and Control file modification, review all types selected for S-JDK. First, fill in each unique type in Table 19. Then, for each type, review the definition in SCLM and Cloud 9 SLR for existence, consistency, and correctness. If the types aren't defined yet, there is another step for updating the project definition and you can include the type definition process there ("Step 4: Update the Project Definition" on page 83).

Note: The following matrix is filled in with the default values.

Type Review Matrix

Table 19. Type Review Matrix

Type	Language	Defined to SCLM	Defined to Cloud 9 SLR	Case Sensitive Yes/No	Binary Yes/No	Lrecl	Recfm
Java	Java			Yes	No	256	VB
Jar	Jar			Yes	Yes	256	VB
Html 1	Ebiz			Yes	No	256	VB
Graphics	Ebiz			Yes	Yes	256	VB
Javaclas	N/A			Yes	Yes	256	VB
Javalist	N/A			Yes	No	256	VB
See "Determining LRECL and file attributes"							

Determining that types exist

To determine if a type exists, go to Cloud 9 and use the List SCLM Files command to access the SCLM Query page. Select a Group then examine the list of Types and verify that the Types are there. If they do not exist, then you must define the Types to SCLM and allocate the data sets in ISPF.

Determining Cloud 9 definitions

To determine if the SCLM type is defined to the Cloud 9 SLR (long name registry), run the IVP JCL member CLZC9J06. Review the output from the job, verifying that the SCLM type has been defined with the proper attributes. If not, modify this job to define the types to the Cloud 9 SLR.

Determining LRECL and file attributes

To determine the Logical Record Length (LRECL) of the type, go into SCLM Option 3.2 and display the data set information for one of the Type data sets. For e-business types, the LRECL is 256 and the RECFM = VB.

Note: If there are records in your e-Business Source types, such as HTML, that extend beyond 256 bytes, you need to assign a library with a larger LRECL

in order to stop truncation. For the purpose of this installation guide, an assumption has been made that your record lengths do not exceed 256 bytes.

Checkpoint #1 for S-JDK for USS installation

At this point the following tasks should be completed.

Table 20. Checkpoint #1 for S-JDK for USS installation

Data Set Names	Completed?
Review all SCLM Inventory Default Values	
Review all USS Directory Default Values	
Determine actual SCLM Inventory and USS Directory Values	
Determine S-JDK Type Matrix	
Document SCLM Types and Cloud 9 definitions.	

Before you begin: S-JDK for USS installation

Chapter 8. Customize translators and translator control files

This part describes the processes to be undertaken in customizing the components that enable you to perform builds and deployments to USS. This involves the modification of JCL members, SCLM translator files and SCLM translator control files.

Step 3a. Review and modify translators

Review and modify CLZTJAVA translator

In this step, you review and modify the CLZTJAVA member found in the **CLZ.SCLZJCL** library. Copy this member to your project definition source and update it to reference the control files listed in "Step 4: Update the Project Definition" on page 83. Ensure that the USS control files are the ones that are specified in the translator at this time.

The Cloud 9 Java Translator uses the following functions: PARSE (save), BUILD, and COPY (promote). The Language is specified as JAVA in the translator, however, if you are going to utilise both the S-JDK for USS build and the S-JDK for RBD build, this can be changed to a different value.

1. The **PARSE function** causes the Java source to be scanned before being saved in SCLM.
2. The **BUILD function** consists of two steps. In Step 1, CLZTRJV1 copies all files contained in the SCLM package that need to be copied to the target server location before invocation of step 2 (the compile step). The UNIXLOC file consists of SCLM-to-USS location mapping rules. The HTTPD file contains EBCDIC-to-ASCII conversion rules.

   The Java compile (JAVAC) is invoked in Step 2, CLZTRJVC, of the BUILD function. The ddname FILEIN contains the source to be compiled. The ddname JCOMPILE contains a shell script that is tailored before invoking the Java compiler. The CLASSPTH file is read and CLASSPATH statements are inserted in the JAVAC shell script. The UNIXLOC SCLM-to-USS rule mapping file is used to send the source, classes, and compiled listing output to the correct location in USS on the mainframe running Cloud 9. When the compile completes, generated class files are written to file CLASSES. The JAVAC compiler output is written to JAVALIST.

   Note: By changing the FLMALLOC for DDNAME=SYSPRINT in the second BUILD step of the JAVA language translator to have a PRINT=Y, the JAVAC output is written out to ddname BLDLIST when a non-zero return code is set. The current statement has PRINT=N.
3. The **COPY function**, CLZTRJVC, deploys Java source to target locations when a SCLM Promote action is invoked. The COPY function can be removed if you do not want to deploy Java source during a Promote action. FILEIN contains the Java source to be deployed. UNIXLOC defines the target location where the source is copied. The HTTPD file contains EBCDIC-to-ASCII conversion rules.

Build Map usage

In the sample translator, CLZTJAVA, Java classes are written to the ddname CLASSES and the listing file is written to the ddname JAVALIST.

For Java class files, more than one Java class file can be created as an output of the compile process. The related SCLM member names are different than the input source member name. Also, the generated Java listing has a different name than the Java source.

The SCLM translator CLZTJAVA uses the IOTYPE=P for the ddnames CLASSES and JAVALIST.

The Build Map for the Java source, as created during the SCLM Build function, contains a list of all Java classes and the name of the Java listing file. These output components of the Java compile process are checked by SCLM only during a SCLM Promote function, but not on a SCLM Build function. Therefore, specifying MODE=CONDITIONAL on a SCLM Build does not cause the Java source to be recompiled when the associated Java class files or listing file are deleted.

Validate the integrity of the Build Map by using the Promote function and specifying MODE=REPORT. If the report shows that your Build Map does not match the Build Map outputs that you have defined, then you need to perform a Build function against the Java source using the MODE=FORCE option.

Control files are described later in this document.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*-------------------------------------------------------------------*
*  NAME:     CLZTJAVA                                               *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE = JAVA, LANGUAGE = JAVA.     *
*-------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR REQUIRES THAT FLMALLOC STATEMENTS BE      *
*         CUSTOMIZED DEPENDING ON WHETHER YOU ARE RUNNING USS OR    *
*         FTP. ALSO VARIOUS INPUT FILES MAY ALSO NEED TO BE         *
*         CUSTOMIZED.  BOTH CONTROL INPUT AND REXX EXECUTABLES ARE  *
*         READ FROM THE PRODUCT LIBRARY CLZ.SCLZCGI.                *
*-------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING   *
*         CONTROL FILES:                                            *
*          CLZ.SCLZCGI(CLZTCPTH)                        USS         *
*          CLZ.SCLZCGI(CLZTCPTW)                        FTP         *
*          CLZ.SCLZCGI(CLZTULOC)                        USS         *
*          CLZ.SCLZCGI(CLZTULOW)                        FTP         *
*          CLZ.SCLZCGI(CLZTHTPD)                                    *
*          CLZ.SCLZCGI(CLZTJAVC)                 USS                *
*          CLZ.SCLZCGI(CLZTJAVW)                 FTP SITE COMMAND   *
*-------------------------------------------------------------------*
*        JAVACLAS AND JAVALIST MUST BE DEFINED AS TYPES TO THE      *
*        PROJDEFS LOAD MODULE.                                      *
*                                                                   *
* CHANGE LOG: Z021120C/A0696 - CHANGE INVOCATION OF REXX            *
*             Z231104A/M0268 - ALTER DDNAME TO DSNAME REFERENCES    *
*-------------------------------------------------------------------*
         FLMLANGL    LANG=JAVA,VERSION=TEXTV1.0
         FLMTRNSL FUNCTN=PARSE,                                     C
               COMPILE=FLMLPGEN,                                    C
               PORDER=1,                                            C
               OPTIONS=(LANG=T,                                     C
               LISTINFO=@@FLMLIS,                                   C
               LISTSIZE=@@FLMSIZ,                                   C
               SOURCEDD=SOURCE,                                     C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
* ------------------------------------------------------------------ *
*                     BUILD TRANSLATOR                              *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                     C
               CALLNAM='COPY PKG MEMBERS',                          C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJV1,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               PDSDATA=Y
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                    USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                    FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
```

Figure 21. CLZTJAVA — Build and Promote S-JDK Translator (Part 1 of 2)

```
* ------------------------------------------------------------------ *
*                     BUILD TRANSLATOR                               *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                        C
               CALLNAM='INVOKE JAVAC',                                 C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJVC,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               OPTIONS='@@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMBR JAC
               VACLAS JAVALIST'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=JCOMPILE,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTJAVC)            USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTJAVW)            FTP SITE COMMAND
         FLMALLOC DDNAME=CLASSPTH,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTCPTH)                      USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTCPTW)                      FTP
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                      USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                      FTP
         FLMALLOC DDNAME=JAVACLAS,DFLTTYP=JAVACLAS,LANG=EBIZ,          C
               LRECL=256,BLKSIZE=27998,RECFM=VB,RECNUM=60000,          C
               IOTYPE=P,KEYREF=OUT1,DIRBLKS=20
         FLMALLOC DDNAME=JAVALIST,DFLTTYP=JAVALIST,LANG=EBIZ,          C
               LRECL=256,BLKSIZE=27998,RECFM=VB,RECNUM=60000,          C
               IOTYPE=P,KEYREF=OUT2
         FLMALLOC DDNAME=SYSPRINT,IOTYPE=O,RECFM=FBA,LRECL=121,        C
               RECNUM=2500,PRINT=N
* ------------------------------------------------------------------ *
*                    PROMOTE TRANSLATOR                              *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=COPY,                                         C
               CALLNAM='JAVA PROMOTE',                                 C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJVP,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               PDSDATA=Y,                                              C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG'
         FLMALLOC DDNAME=SYSEXEC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                    USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                    FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ------------------------------------------------------------------ *
```

Figure 21. CLZTJAVA — Build and Promote S-JDK Translator (Part 2 of 2)

Review and modify CLZTJAR translator

In this step, you review and modify the CLZTJAR member found in the **CLZ.SCLZJCL** library delivered with the SMP/E installation of Cloud 9. Copy this member to your project definition source and update it to reference the control files listed in "Step 4: Update the Project Definition" on page 83. Ensure that the USS control files are the ones that are specified in the translator at this time.

The Cloud 9 Jar Translator uses the following functions: PARSE (save), BUILD, and COPY (promote). The Language is specified as JAR in the translator, however, if you are going to utilise both the Jar USS build and the Jar FTP build, this can be changed to a different value.

The **PARSE function** causes Cloud 9 Jar control statements to be scanned before being saved in SCLM.

The **BUILD function**, CLZTRJAR, invokes the Jar compiler. The ddname FILEIN specifies the Jar directives specifying the location of the files to be included in the Jar. The ddname JARCOMP contains a shell script that is tailored before invoking the Jar compiler. The UNIXLOC SCLM-to-USS rule mapping file is used to send the tailored shell to USS. This file defines the location of the input files and the listing output for this Jar compile. When the compile completes, the generated jar file is written to file JAR. The Jar compiler output is written to JARLIST.

The COPY function, CLZTRJVC, deploys Jar source to target locations when a SCLM Promote action is invoked. FILEIN contains the Jar file to be deployed. UNIXLOC is defines the target location where the source is copied. The HTTPD file contains EBCDIC-to-ASCII conversion rules.

Control files are described later in this document.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*--------------------------------------------------------------------*
*  NAME:     CLZTJAR                                                 *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE = JAR, LANGUAGE = JAR.        *
*--------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR REQUIRES THAT FLMALLOC STATEMENTS BE       *
*         CUSTOMIZED DEPENDING ON WHETHER YOU ARE RUNNING USS OR     *
*         FTP. ALSO VARIOUS INPUT FILES MAY ALSO NEED TO BE          *
*         CUSTOMIZED.  BOTH CONTROL INPUT AND REXX EXECUTABLES ARE   *
*         READ FROM PRODUCT LIBRARY CLZ.SCLZCGI.                     *
*--------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING    *
*         CONTROL FILES:                                             *
*          CLZ.SCLZCGI(CLZTULOC)                          USS        *
*          CLZ.SCLZCGI(CLZTULOW)                          FTP        *
*          CLZ.SCLZCGI(CLZTHTPD)                                     *
*          CLZ.SCLZCGI(CLZTJARU)                          USS        *
*          CLZ.SCLZCGI(CLZTJARW)                          FTP        *
*--------------------------------------------------------------------*
*        JAR AND JARLIST MUST BE DEFINED AS TYPES TO THE             *
*        PROJDEFS LOAD MODULE.                                       *
*                                                                    *
* CORRECTION: Z021120C/A0696 - CHANGE INVOCATION OF REXX             *
* D RICHARD : Z230825A/A0960 - CHANGE ALLOCATION OF JARLIST          *
* D RICHARD : Z230915A/Z230825A - REWORK WITH CORRECT VERSION        *
```

Figure 22. CLZTJAR — Build and Promote JAR S-JDK Translator (Part 1 of 2)

```
*-------------------------------------------------------------------*
         FLMLANGL    LANG=JAR,VERSION=TEXTV1.0
         FLMTRNSL FUNCTN=PARSE,                                     C
               COMPILE=FLMLPGEN,                                    C
               PORDER=1,                                            C
               OPTIONS=(LANG=T,                                     C
               LISTINFO=@@FLMLIS,                                   C
               LISTSIZE=@@FLMSIZ,                                   C
               SOURCEDD=SOURCE,                                     C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
* ----------------------------------------------------------------- *
*                     BUILD TRANSLATOR                              *
* ----------------------------------------------------------------- *
         FLMTRNSL FUNCTN=BUILD,                                     C
               CALLNAM='BUILD JAR',                                 C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJAR,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               OPTIONS='@@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMBR JAC
               R JARLIST'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=JARCOMP,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTJARU)                       USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTJARW)                       FTP
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                       USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                       FTP
         FLMALLOC DDNAME=JAR,IOTYPE=P,PRINT=Y,RECNUM=60000,         C
               RECFM=VB,BLKSIZE=27998,                              C
               DFLTTYP=JAR,KEYREF=OBJ,LRECL=256,LANG=EBIZ
         FLMALLOC DDNAME=JARLIST,DFLTTYP=JARLIST,LANG=EBIZ,         C
               LRECL=256,BLKSIZE=27998,RECFM=VB,RECNUM=60000,       C
               IOTYPE=P,KEYREF=LIST
         FLMALLOC DDNAME=SYSPRINT,IOTYPE=O,RECFM=FBA,LRECL=121,     C
               RECNUM=2500,PRINT=N
* ----------------------------------------------------------------- *
*                    PROMOTE TRANSLATOR                             *
* ----------------------------------------------------------------- *
         FLMTRNSL FUNCTN=COPY,                                      C
               CALLNAM='PROMOTE JAR',                               C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJVP,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               PDSDATA=Y,                                           C
               OPTIONS='BINARY @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLC
               MMBR @@FLMTOG'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                       USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                       FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ----------------------------------------------------------------- *
```

Figure 22. CLZTJAR — Build and Promote JAR S-JDK Translator (Part 2 of 2)

Using the JAR translator

The Cloud 9 Jar control statements are stored in SCLM in a JARMAKE member. This member should include a file extension of either .CMD or .BAT (This is dependent upon the target operating system, WINDOWS=.BAT, USS=.CMD). There are two supported variations of this standard:

- JARMAKEFILE.JAR.CMD (or .BAT) - In this variation, the .CMD/.BAT portion will be removed to create the resulting JAR name (JARMAKEFILE.JAR)
- JARMAKEFILE.CMD (or .BAT) - In this variation, the .CMD/.BAT portion will be overlaid with .JAR to create the resulting JAR name (JARMAKEFILE.JAR)

The invocation member (CLZTRJAR) in the SCLZCGI library may now pass an additional option (KEEPCMD=YES|NO) to CLZTLJAR. If KEEPCMD=NO (the default action), the temporary command file used to create the jar is deleted after it has been used. If KEEPCMD=NO, the temporary command file is retained on the remote system. The option should only be used for debugging purposes, as the command file is not in the build map.

Sample Jar make file

The JARMAKE member can use either absolute pathing or relative pathing (relative from location of the JAR file, which depends upon the directory that the remote build translator deploys the JARMAKE member. For example, given this directory tree:

```
/DEV
    |
    /CLASSES
    /GRAPHICS
    /JAVA
```

and that the resulting JAR file is to be stored in /DEV/JAVA, the contents of the JARMAKE member could be:

```
/DEV/CLASSES/Classname.CLASS          (Absolute path)
../CLASSES/ Class2.CLASS              (Relative path)
../GRAPHICS/Picture.GIF               (Relative path)
```

Another example of the jar process is to create a jar file using a combination of locations and specific files. Our sample file will build a jar file composed of the contents of several relative directory structures. The directory structures are relative to the directory where the jarmake file is placed. This location is controlled by the contents of the file associated with the unixloc ddname. For our example, we have associated the following file with an SCLM type of JARMAKE and a language of JARMAKE. The file has a name of TESTME.BAT. The name of the jar file created by this process will be the member name with the extension of .jar overlaid on it. For this example, that means the jar name will be TESTME.JAR.

The TESTME.BAT contents:

```
org\apache\bcel
org\IBM\install
graphics\mouse.jpg
control\stuff.htm
```

The JARMAKE file has four lines that all begin in column 1. The jar will be composed of the directory structure `org\apache\bcel` and all of its sub-directories; `org\IBM\install` and its sub-directories; and the two individual files, `graphics\mouse.jpg`, and `control\stuff.htm`.

Review and modify e-Business translator CLZTJBIZ

In this step, you review and modify the CLZTJBIZ member found in the **CLZ.SCLZJCL** library delivered with the SMP/E installation of Cloud 9. Copy this member to your project definition source and update it to reference the control files listed in "Step 4: Update the Project Definition" on page 83. Ensure that the USS control files are the ones that are specified in the translator at this time.

Note: With APAR OW56539, translator CLZTJBIZ replaces the translators CLZTJBIN and CLZTJTXT through the use of the UNIXLOC file. However CLZTJBIN and CLZTJTXT are both still supported.

The Cloud 9 e-Business translator uses the following functions: PARSE (save), BUILD and COPY (promote). The Language of EBIZ is specified in the translator.

The **PARSE function** causes SCLM to parse input before being saving e-business objects in SCLM.

The **BUILD and COPY function**, CLZTRJVP, invokes the Cloud 9 e-business deployment program. The ddname FILEIN contains the file to be deployed. The UNIXLOC file consists of the SCLM-to-USS location mapping rules. The HTTPD file contains EBCDIC-to-ASCII conversion rules. File extensions associated with files added with the language EBIZ must be defined to the UNIXLOC file. If the file extension is not found in the UNIXLOC file, then parameter 2 of this translator specifies the default to be used by this translator.

Control files are described later in this document.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*--------------------------------------------------------------------*
*  NAME:     CLZTJBIZ    (CLZ.SCLZJCL)                               *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE = GRAPHICS, HTML, XML        *
*            LANGUAGE = EBIZ.                                        *
*--------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR DOES NOT REQUIRE CUSTOMIZATION, HOWEVER,  *
*         THE VARIOUS INPUT FILES DO. BOTH CONTROL INPUT AND REXX   *
*         EXECUTABLE CODE ARE READ IN FROM THE PRODUCT LIBRARY      *
*         CLZ.SCLZCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECS.    *
*--------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING   *
*         CONTROL FILES:                                            *
*          CLZ.SCLZCGI(CLZTULOC)                          USS       *
*          CLZ.SCLZCGI(CLZTULOW)                          FTP       *
*          CLZ.SCLZCGI(CLZTHTPD)                                    *
*--------------------------------------------------------------------*
         FLMLANGL    LANG=EBIZ,VERSION=TEXTV1.0
```

Figure 23. CLZTBIZ — Build and Promote BIZ S-JDK Translator (Part 1 of 2)

```
         FLMTRNSL FUNCTN=PARSE,                                        C
               COMPILE=FLMLPGEN,                                       C
               PORDER=1,                                               C
               OPTIONS=(LANG=T,                                        C
               LISTINFO=@@FLMLIS,                                      C
               LISTSIZE=@@FLMSIZ,                                      C
               SOURCEDD=SOURCE,                                        C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
* ------------------------------------------------------------------ *
*                   BUILD TRANSLATOR                                 *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                        C
               CALLNAM='UNIX BUILD',                                   C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJVP,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG @@FLMBIO'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                      USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                      FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ------------------------------------------------------------------ *
*                   PROMOTE  TRANSLATOR                              *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=COPY,                                         C
               CALLNAM='UNIX PROMOTE',                                 C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJVP,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               PDSDATA=Y,                                              C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                      USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                      FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ------------------------------------------------------------------ *
```

Figure 23. CLZTBIZ — Build and Promote BIZ S-JDK Translator (Part 2 of 2)

Review and modify CLZTJTXT translator

Note: With APAR OW56539, translator CLZTJBIZ replaces the translators CLZTJBIN and CLZTJTXT through the use of the UNIXLOC file. However CLZTJBIN and CLZTJTXT are both still supported.

In this step, you review and modify the CLZTJTXT member found in the **CLZ.SCLZJCL** library delivered with the SMP/E installation of Cloud 9.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*-------------------------------------------------------------------*
*  NAME:     CLZTJTXT                                               *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE=HTML,LANGUAGE=EBIZTEXT       *
*-------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR DOES NOT REQUIRE CUSTOMIZATION, HOWEVER,  *
*         THE VARIOUS INPUT FILES DO. BOTH CONTROL INPUT AND REXX   *
*         EXECUTABLE CODE ARE READ IN FROM THE PRODUCT LIBRARY      *
*         CLZ.SCLZCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECS.    *
*-------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING   *
*         CONTROL FILES:                                            *
*          CLZ.SCLZCGI(CLZTULOC)                                    *
*-------------------------------------------------------------------*
         FLMLANGL    LANG=EBIZTEXT,VERSION=TEXTV1.0
         FLMTRNSL FUNCTN=PARSE,                                     C
               COMPILE=FLMLPGEN,                                    C
               PORDER=1,                                            C
               OPTIONS=(LANG=T,                                     C
               LISTINFO=@@FLMLIS,                                   C
               LISTSIZE=@@FLMSIZ,                                   C
               SOURCEDD=SOURCE,                                     C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
* ------------------------------------------------------------------ *
*                   BUILD TRANSLATOR                                 *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                     C
               CALLNAM='UNIX BUILD',                                C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJVP,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG @@FLMBIO'
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)
         FLMALLOC DDNAME=WORKFILE,IOTYPE=W,LRECL=32000,RECFM=V
* ------------------------------------------------------------------ *
*                   PROMOTE  TRANSLATOR                              *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=COPY,                                      C
               CALLNAM='UNIX PROMOTE',                              C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJVP,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               PDSDATA=Y,                                           C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG'
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)
         FLMALLOC DDNAME=WORKFILE,IOTYPE=W,LRECL=32000,RECFM=V
* ------------------------------------------------------------------ *
```

Figure 24. CLZTJTXT — Build and Promote Text S-JDK Translator

Review and modify CLZTJBIN translator

Note: With APAR OW56539, translator CLZTJBIZ replaces the translators CLZTJBIN and CLZTJTXT through the use of the UNIXLOC file. However CLZTJBIN and CLZTJTXT are both still supported.

In this step, you review and modify the CLZTJBIN member found in the **CLZ.SCLZJCL** library delivered with the SMP/E installation of Cloud 9.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*--------------------------------------------------------------------*
*  NAME:     CLZTJBIN                                                *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE=GRAPHICS,LANGUAGE=EBIZBIN     *
*--------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR DOES NOT REQUIRE CUSTOMIZATION, HOWEVER,   *
*         THE VARIOUS INPUT FILES DO. BOTH CONTROL INPUT AND REXX    *
*         EXECUTABLE CODE ARE READ IN FROM THE PRODUCT LIBRARY       *
*         CLZ.SCLZCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECS.     *
*--------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING    *
*         CONTROL FILES:                                             *
*          CLZ.SCLZCGI(CLZTULOC)                                     *
*--------------------------------------------------------------------*
         FLMLANGL    LANG=EBIZBIN,VERSION=TEXTV1.0
         FLMTRNSL FUNCTN=PARSE,                                      C
               COMPILE=FLMLPGEN,                                     C
               PORDER=1,                                             C
               OPTIONS=(LANG=T,                                      C
               LISTINFO=@@FLMLIS,                                    C
               LISTSIZE=@@FLMSIZ,                                    C
               SOURCEDD=SOURCE,                                      C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
* ------------------------------------------------------------------ *
*                  BUILD TRANSLATOR                                  *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                       C
               CALLNAM='UNIX BUILD',                                  C
               CALLMETH=TSOLNK,                                       C
               COMPILE=CLZTRJVP,                                      C
               DSNAME=CLZ.SCLZCGI,                                    C
               OPTIONS='BINARY @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLC
               MBR @@FLMTOG'
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)
         FLMALLOC DDNAME=WORKFILE,IOTYPE=W,LRECL=32000,RECFM=V
```

Figure 25. CLZTJBIN — Build and Promote Binary S-JDK Translator (Part 1 of 2)

```
* ------------------------------------------------------------------ *
*                  PROMOTE  TRANSLATOR                               *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=COPY,                                        C
               CALLNAM='UNIX PROMOTE',                                C
               CALLMETH=TSOLNK,                                       C
               COMPILE=CLZTRJVP,                                      C
               DSNAME=CLZ.SCLZCGI,                                    C
               PDSDATA=Y,                                             C
               OPTIONS='BINARY @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLC
               MMBR @@FLMTOG'
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)
         FLMALLOC DDNAME=WORKFILE,IOTYPE=W,LRECL=32000,RECFM=V
* ------------------------------------------------------------------ *
```

Figure 25. CLZTJBIN — Build and Promote Binary S-JDK Translator (Part 2 of 2)

Step 3b: Review and modify translator control files

The translator tailoring in this section uses the SCLM project IBMDEMO, along with it's groups and types, as an example. You can change these translator control files to contain your own SCLM project with the groups and types that you have defined.

CASE SENSITIVITY ALERT!

The following translator control files contain case sensitive data. To ensure that the values not automatically changed to uppercase during editing, issue the 'CAPS OFF' command on the command line of your ISPF session.

Modify CLZTULOC — common SCLM to USS Life Cycle Mapping Rules

In this step, you review and modify the CLZTULOC member found in the SCLZCGI library.

This member is input to all S-JDK translators. It is used to map the SCLM to USS life cycle locations. This control file is used in the Build, Promote and Delete process.

This control member provides several functions to Cloud 9 translators. First, it defines the SCLM to USS mapping. Second, it is used to set file access permissions for files sent to USS. Third, it defines whether files are to be deleted from their source locations (source and target) when an SCLM Promote is requested.

```
* ------------------------------------------------------------- *
* CLOUD 9 JAVA/USS S-JDK COMPONENT                              *
* ------------------------------------------------------------- *
* NAME:    CLZTULOC                                             *
* PURPOSE: SCLM TO UNIX LIFE CYCLE MAPPING RULES.               *
* REFER:   DIRECTLY REFERENCED IN TRANSLATOR DDNAME UNIXLOC.    *
* ------------------------------------------------------------- *
* prj,alt,grp,typ            unix location     KEEP or DELETE
*                                                on promote  Permissions
IBMDEMO,IBMDEMO,DEV,GRAPHICS /u/ibmdemo/dev/graphics  KEEP   PERM=777
IBMDEMO,IBMDEMO,DEV,HTML     /u/ibmdemo/dev           KEEP   PERM=777
IBMDEMO,IBMDEMO,DEV,HTML     /u/ibmdemo/dev/html      KEEP   PERM=775
IBMDEMO,IBMDEMO,DEV,JAVA     /u/ibmdemo/dev           KEEP   PERM=777
IBMDEMO,IBMDEMO,DEV,JAVACLAS /u/ibmdemo/dev/classes   KEEP   PERM=775
IBMDEMO,IBMDEMO,DEV,JAR      /u/ibmdemo/dev/classes   KEEP   PERM=775
IBMDEMO,IBMDEMO,DEV,JAVALIST /u/ibmdemo/dev/listings  KEEP   PERM=775
IBMDEMO,IBMDEMO,DEV,JARLIST  /u/ibmdemo/dev/listings  KEEP   PERM=775
*
IBMDEMO,IBMDEMO,QA,GRAPHICS  /u/ibmdemo/qa/graphics   KEEP
IBMDEMO,IBMDEMO,QA,HTML      /u/ibmdemo/qa            KEEP
IBMDEMO,IBMDEMO,QA,HTML      /u/ibmdemo/qa/html       KEEP
IBMDEMO,IBMDEMO,QA,JAVA      /u/ibmdemo/qa            KEEP
IBMDEMO,IBMDEMO,QA,JAVACLAS  /u/ibmdemo/qa/classes    KEEP
IBMDEMO,IBMDEMO,QA,JAR       /u/ibmdemo/qa/classes    KEEP
IBMDEMO,IBMDEMO,QA,JAVALIST  /u/ibmdemo/qa/listings   KEEP
IBMDEMO,IBMDEMO,QA,JARLIST   /u/ibmdemo/qa/listings   KEEP
*
IBMDEMO,IBMDEMO,REL,GRAPHICS /u/ibmdemo/rel/graphics  KEEP
IBMDEMO,IBMDEMO,REL,HTML     /u/ibmdemo/rel           KEEP
IBMDEMO,IBMDEMO,REL,HTML     /u/ibmdemo/rel/html      KEEP
IBMDEMO,IBMDEMO,REL,JAVA     /u/ibmdemo/rel           KEEP
IBMDEMO,IBMDEMO,REL,JAVACLAS /u/ibmdemo/rel/classes   KEEP
IBMDEMO,IBMDEMO,REL,JAR      /u/ibmdemo/rel/classes   KEEP
IBMDEMO,IBMDEMO,REL,JAVALIST /u/ibmdemo/rel/listings  KEEP
IBMDEMO,IBMDEMO,REL,JARLIST  /u/ibmdemo/rel/listings  KEEP
```

Figure 26. CLZTULOC — Common SCLM to USS Life Cycle Map

Position 1
SCLM Life Cycle location containing project, alternative project, group and type

Position 2
UNIX System Services Life Cycle location

Position 3
Dispostion of files on promote

Position 4
Permissions to be allocated to files created in the specified directory. This parameter is used to give different UNIX permissions to files of different types, and also to files at different levels in the hierarchy.

Modify CLZTCPTH — common %CLASSPATH% substitution

In this step, you review and modify the CLZTCPTH member found in the SCLZCGI library.

This member is input to both the Java and Jar compile shells. It is used to fill in the %Classpath% variable in the compiler shells.

```
* ------------------------------------------------------------- *
*  CLOUD 9 JAVA/USS S-JDK COMPONENT                             *
* ------------------------------------------------------------- *
* NAME:    CLZTCPTH                                            *
* PURPOSE: %CLASSPATH% SUBSTITUTION FILE.                      *
* REFER:   DIRECTLY REFERENCED IN TRANSLATOR DDNAME CLASSPTH   *
* ------------------------------------------------------------- *
* prj,alt,gr        classpath and java source concatenation
IBMDEMO,IBMDEMO,DEV /u/ibmdemo/dev/classes
IBMDEMO,IBMDEMO,DEV /u/ibmdemo/dev/java
IBMDEMO,IBMDEMO,DEV /u/ibmdemo/qa/classes
IBMDEMO,IBMDEMO,DEV /u/ibmdemo/qa/java
IBMDEMO,IBMDEMO,DEV /u/ibmdemo/rel/classes
IBMDEMO,IBMDEMO,DEV /u/ibmdemo/rel/java
```

Figure 27. CLZTCPTH — Common %CLASSPATH% Substitution

This is the SCLM hierarchy class path allocation control member. It tells the Java compile where to find additional class files that are within the SCLM hierarchy. Java source libraries are included in the concatenation because that is the way Java works. If Java finds a class file it doesn't have in the class directory, it looks in the source directory for the Java source. The Java compiler then compiles the source to create the required class file. When the source directory is included as part of the class path allocation, Java can find the source, even in cases where the class file is not in the class path.

Note: The actual class path might be more complex than one shown in Figure 27. For instance, the core Java class libraries might reside in directories outside of the standard SCLM /USS life cycle.

Position 1
SCLM Life Cycle location at which a build is occuring

Position 2
UNIX System Services class location. This contains the complete hierarchy required to find the class files during a compile.

Modify CLZTJAVC — Java compile shell

In this step, you review and modify the CLZTJAVC member found in the SCLZCGI library.

This member is input to the CLZTJAVA translator for the Java Type. (You might need to review the path location with your USS System Programmer.) This file is the template used to invoke the Java compile. The template is modified with the %CLASSPATH% statement, which is substituted with data taken from CLZTCPTH. The parameters %1, %2 and %3 are substituted, based on rules defined in the CLZTULOC file as follows:

- %1 = java source location
- %2 = java file name
- %3 = java class location

The symbolic parms denoted by %1, %2, and %3 can represent long directory names and long file names. To create the correct syntax when these long names contain spaces or apostrophes, the symbolics should be surrounded by quotation marks, for example:

```
"%1" "%2" "%3"
```

This syntax can also be used when the directory and file names do not contain spaces or apostrophes.

```
# ------------------------------------------------------------- #
# CLOUD 9 JAVA/USS S-JDK COMPONENT                              #
# ------------------------------------------------------------- #
# NAME:    CLZTJAVC                                             #
# PURPOSE: JAVA COMPILE SHELL                                   #
# REFER:   DIRECTLY REFERENCED IN THE TRANSLATOR DDNAME JCOMPILE. #
# ------------------------------------------------------------- #
export PATH=/usr/lpp/java/J1.1/bin:$PATH
export CLASSPATH=%CLASSPATH%:$CLASSPATH
cd %1
javac -verbose -d %3 %2
```

Figure 28. CLZTJAVC — Java Compile Shell

Note: The actual class path might be more complex than one shown in Figure 28. For instance, the core Java class libraries might reside in directories outside of the standard SCLM /USS life cycle.

Modify CLZTJARU — Jar compile shell

In this step, you review and modify the CLZTJARU member found in the SCLZCGI library. This member is the default input to the CLZTJAR translator for the JAR Type. The template is modified with the #CLASSPATH# statement substituted with data taken from CLZTCPTH (USS). The Jar commands are appended to the template before invocation of the Jar compiler.

```
# ------------------------------------------------------------- #
# CLOUD 9 JAVA/USS S-JDK COMPONENT                              #
# ------------------------------------------------------------- #
# NAME:    CLZTJARU                                             #
# PURPOSE: JAR COMPILE SHELL                                    #
# REFER:   DIRECTLY REFERENCED IN TRANSLATOR DDNAME JARCOMP.    #
# ------------------------------------------------------------- #
export JAVA_HOME=/usr/lpp/java/J1.1
export PATH=/usr/lpp/java/J1.1/bin:$PATH
```

Figure 29. CLZTJARU — JAR Compile Shell

Modify CLZTHTPD — Addtype list for Java compile

In this step, you review and modify the CLZTHTPD member found in the SCLZCGI target library.

This member contains Addtype definitions, similar to those that can be found in the CLZHTTPD member in the SCLZHTML target library. Addtype definitions are used by the Java compile process to determine if objects included in the Java compile are Binary or Text. This is required by the copy function that is part of the compile process.

```
#------------------------------------------------------------------------
#  Name:      CLZTHTPD
#  Purpose:   Cloud9 Server Rules File
#  Usage:     This file is used by the Java compile process
#------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------------
#
#Non-standard MIME types declared here.  (User style MIME types)
#
#-------------------------------------------------------------------------------------------------
AddType  .asm      text/asm                          ebcdic  1.0 # Assemble Macros
AddType  .doc      binary/doc                        binary  1.0 # Microsoft Word Documents
AddType  .ppt      binary/ppt                        binary  1.0 # Power Point Documents
AddType  .cob      text/cobol                        ebcdic  1.0 # COBOL Source Code
AddType  .cbl      text/cobol                        ebcdic  1.0 # COBOL Source Code
AddType  .cobol    text/cobol                        ebcdic  1.0 # COBOL Source Code
#-------------------------------------------------------------------------------------------------
#
AddType  .cer      application/x-x509-user-cert      ebcdic  0.5 # Browser Certificate
AddType  .der      application/x-x509-ca-cert        binary  1.0 # CA Certificate
AddType  .mime     www/mime                          binary  1.0 # Internal -- MIME is
AddType  .bin      application/octet-stream          binary  1.0 # Uninterpreted binary
AddType  .class    application/octet-stream          binary  1.0 # Java applet or application
AddType  .pdf      application/pdf                   binary  1.0
AddType  .ai       application/postscript            ebcdic  0.5 # Adobe Illustrator
AddType  .PS       application/postscript            ebcdic  0.8 # PostScript
AddType  .eps      application/postscript            ebcdic  0.8
AddType  .ps       application/postscript            ebcdic  0.8
AddType  .rtf      application/x-rtf                 ebcdic  1.0 # RTF
AddType  .csh      application/x-csh                 ebcdic  0.5 # C-shell script
AddType  .latex    application/x-latex               ebcdic  1.0 # LaTeX source
AddType  .cdf      application/x-cdf                 ebcdic  1.0 # Channel Definition Format
AddType  .sh       application/x-sh                  ebcdic  0.5 # Shell-script
AddType  .tcl      application/x-tcl                 ebcdic  0.5 # TCL-script
AddType  .tex      application/x-tex                 ebcdic  1.0 # TeX source
AddType  .t        application/x-troff               ebcdic  0.5 # Troff
AddType  .roff     application/x-troff               ebcdic  0.5
AddType  .tr       application/x-troff               ebcdic  0.5
AddType  .man      application/x-troff-man           ebcdic  0.5 # Troff with man macros
AddType  .me       application/x-troff-me            ebcdic  0.5 # Troff with me macros
AddType  .ms       application/x-troff-ms            ebcdic  0.5 # Troff with ms macros
AddType  .gtar     application/x-gtar                binary  1.0 # Gnu tar
AddType  .shar     application/x-shar                ebcdic  1.0 # Shell archive
AddType  .wrl      x-world/x-vrml                   binary  1.0 # VRML
AddType  .snd      audio/basic                      binary  1.0 # Audio
AddType  .au       audio/basic                      binary  1.0
AddType  .aiff     audio/x-aiff                     binary  1.0
AddType  .aifc     audio/x-aiff                     binary  1.0
AddType  .aif      audio/x-aiff                     binary  1.0
AddType  .wav      audio/x-wav                      binary  1.0 # Windows+ WAVE format
AddType  .bmp      image/bmp                        binary  1.0 # OS/2 bitmap format
AddType  .gif      image/gif                        binary  1.0 # GIF
AddType  .ief      image/ief                        binary  1.0 # Image Exchange fmt
AddType  .jpg      image/jpeg                       binary  1.0 # JPEG
AddType  .JPG      image/jpeg                       binary  1.0
AddType  .JPE      image/jpeg                       binary  1.0
AddType  .jpe      image/jpeg                       binary  1.0
AddType  .JPEG     image/jpeg                       binary  1.0
AddType  .jpeg     image/jpeg                       binary  1.0
AddType  .tif      image/tiff                       binary  1.0 # TIFF
AddType  .tiff     image/tiff                       binary  1.0
AddType  .ras      image/cmu-raster                 binary  1.0
AddType  .pnm      image/x-portable-anymap          binary  1.0 # PBM Anymap format
AddType  .pbm      image/x-portable-bitmap          binary  1.0 # PBM Bitmap format
AddType  .pgm      image/x-portable-graymap         binary  1.0 # PBM Graymap format
AddType  .ppm      image/x-portable-pixmap          binary  1.0 # PBM Pixmap format
AddType  .rgb      image/x-rgb                      binary  1.0
AddType  .xbm      image/x-xbitmap                  ebcdic  1.0 # X bitmap
AddType  .xpm      image/x-xpixmap                  binary  1.0 # X pixmap format
AddType  .xwd      image/x-xwindowdump              binary  1.0 # X window dump (xwd)
AddType  .html     text/html                        ebcdic  1.0 # HTML
AddType  .htm      text/html                        ebcdic  1.0 # HTML on PCs
AddType  .htmls    text/x-ssi-html                  ebcdic  1.0 # Server-side includes
AddType  .shtml    text/x-ssi-html                  ebcdic  1.0 # Server-side includes
```

Figure 30. CLZTHTPD — Cloud 9 Server Rules (Part 1 of 2)

```
AddType  .c        text/plain                      ebcdic  0.5 # C source
AddType  .h        text/plain                      ebcdic  0.5 # C headers
AddType  .C        text/plain                      ebcdic  0.5 # C++ source
AddType  .cc       text/plain                      ebcdic  0.5 # C++ source
AddType  .hh       text/plain                      ebcdic  0.5 # C++ headers
AddType  .java     text/plain                      ebcdic  0.5 # Java source
AddType  .js       text/plain                      ebcdic  0.5 # JavaScript source
AddType  .m        text/plain                      ebcdic  0.5 # Objective-C source
AddType  .f90      text/plain                      ebcdic  0.5 # Fortran 90 source
AddType  .txt      text/plain                      ebcdic  0.5 # Plain text
AddType  .bat      text/plain                      ebcdic  0.5 # Plain text
AddType  .css      text/css                        8bit    1.0 # W3C Cascading Style Sheets
AddType  .rtx      text/richtext                   ebcdic  1.0 # MIME Richtext format
AddType  .tsv      text/tab-separated-values       ebcdic  1.0 # Tab-separated values
AddType  .etx      text/x-setext                   ebcdic  0.9 # Struct Enchanced Txt
AddType  .MPG      video/mpeg                      binary  1.0 # MPEG
AddType  .mpg      video/mpeg                      binary  1.0
AddType  .MPE      video/mpeg                      binary  1.0
AddType  .mpe      video/mpeg                      binary  1.0
AddType  .MPEG     video/mpeg                      binary  1.0
AddType  .mpeg     video/mpeg                      binary  1.0
AddType  .qt       video/quicktime                 binary  1.0 # QuickTime
AddType  .mov      video/quicktime                 binary  1.0
AddType  .avi      video/x-msvideo                 binary  1.0 # MS Video for Windows
AddType  .movie    video/x-sgi-movie               binary  1.0 # SGI  moviepalyer
AddType  .zip      multipart/x-zip                 binary  1.0 # PKZIP
AddType  .tar      multipart/x-tar                 binary  1.0 # 4.3BSD tar
AddType  .ustar    multipart/x-ustar               binary  1.0 # POSIX tar
AddType  *.*       www/unknown                     binary  0.2 # Try to guess
AddType  *         www/unknown                     binary  0.2 # Try to guess
AddType  .cxx      text/plain                      ebcdic  0.5 # C++
AddType  .for      text/plain                      ebcdic  0.5 # Fortran
AddType  .mar      text/plain                      ebcdic  0.5 # MACRO
AddType  .log      text/plain                      ebcdic  0.5 # logfiles
AddType  .com      text/plain                      ebcdic  0.5 # scripts
AddType  .sdml     text/plain                      ebcdic  0.5 # SDML
AddType  .list     text/plain                      ebcdic  0.5 # listfiles
AddType  .lst      text/plain                      ebcdic  0.5 # listfiles
AddType  .def      text/plain                      ebcdic  0.5 # definition files
AddType  .conf     text/plain                      ebcdic  0.5 # definition files
AddType  .         text/plain                      ebcdic  0.5 # files with no extension
AddType  .JP932    text/x-DBCS                     binary  1.0 IBM-932  # Japanese DBCS
AddType  .JPeuc    text/x-DBCS                     binary  1.0 IBMeucJP # Japanese DBCS
```

Figure 30. CLZTHTPD — Cloud 9 Server Rules (Part 2 of 2)

Position 1
Addtype keyword

Position 2
file extension

Position 3
MIME type and subtype you want to bind to files that match the corresponding suffix pattern.

Position 4
The MIME content encoding to which the data has been converted.

Position 5
An optional indicator of relative value (on a scale of 0.0 to 1.0) for the content type.

Position 6
Description to associate with the document

For more information about the Addtype directives see the *HTTP Server Planning, Installing, and Using* manual (SC34-4826-00).

Java tracing

The tracing facility for the Java process aids with debugging problems. It can be helpful to see if the tailoring of the Java control members has been carried out properly. The tracing can be turned on in the following modules, found in the SCLZCGI target library: CLZTRJAR, CLZTRJVC, CLZTRJVP, CLZTRJV1 and CLZTRJDL. Each of these members contains two calls to the same program, one of which uses the DEBUG parameter and is commented out. To activate the tracing, change the commenting to use the call with the DEBUG parameter. The default is to have tracing turned off.

Checkpoint #2 for S-JDK for USS installation

At this point you should have completed the following tasks.

Table 21. Checkpoint #2 for S-JDK for USS installation

Task	Completed?
Reviewed and modified JAVA translator CLZTJAVA	
Reviewed and modified JAR translator CLZTJAR	
Reviewed and modified EBIZ translator CLZTJBIZ	
Reviewed and modified TEXT translator CLZTJTXT (optional)	
Reviewed and modified GRAPHICS translator CLZTJBIN (optional)	
Reviewed and modified the USS to SCLM mapping control file CLZTULOC	
Reviewed and modified class path control file CLZTCPTH	
Reviewed and modified JAVA compile shell CLZTJAVC	
Reviewed and modified JAR compile shell CLZTJARU	

Chapter 9. Define the S-JDK inventory, USS, and Cloud 9 parts

Step 4: Update the Project Definition

To complete the enabling of the S-JDK, the current SCLM Project Definition must be updated with both the S-JDK types and the S-JDK translators. These definitions can be included with the current project definitions that define your existing project or used as a standalone SCLM Project. The following section shows the type definitions and translator copy statements required to define the S-JDK defaults. Typically, these types and translators are included in a common project. The following member shows a stand-alone project definition JCL stream containing the Type and translator definitions.

The following member JCL member, CLZTPDEF, can be found in the CLZ.SCLZJCL library. This is meant as an example. With your SCLM administrator, you need to determine if the new types will be an isolated project or part of an existing project.

Those lines that might have to be added to your own project definitions are identified by being highlighted in Bold.

```
//* (JOBCARD)
//*--------------------------------------------------------------------*
//*  NAME:     CLZTPDEF                                                *
//*  PURPOSE:  S-JDK STANDALONE PROJECT DEFINITION.                    *
//*--------------------------------------------------------------------*
//*  TO USE THIS JCL YOU MUST:                                         *
//*         1) INSERT A VALID JOB CARD.                                *
//*         2) REVIEW THE SCLM VALUES - THEY ARE INITIALLY SET TO      *
//*            THE S-JDK DEFAULT VALUES.                               * 0
//*         3) REVIEW THE DATASET NAMES - THEY ARE INITIALLY SET TO    *
//*            THE S-JDK DEFAULT VALUES.                               *
//*         4) MODIFY THE DATASET NAMES TO MATCH ACTUAL CHOSEN S-JDK   *
//*            TYPES.                                                  *
//*         4) MODIFY THE SCLM VALUES TO MATCH ACTUAL CHOSEN S-JDK     *
//*            TYPES.                                                  *
//*         5) REVIEW AND MODIFY THE S-JDK TRANSLATORS AND LANGUAGE    *
//*            NAMES TO THE CHOSE TYPES.                               *
//*         6) CHANGE THE UNIT=TDISK TO THE APPROPRIATE UNIT FOR       *
//*            TEMPORARY FILES.                                        *
//COMP   PROC
//*--------------------------------------------------------------------*
//STEP1ASM EXEC PGM=ASMA90,REGION=3072K,COND=(0,NE),
//             PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT)'
//SYSLIB   DD  DSN=CLZ.SCLZJCL,DISP=SHR
//         DD  DSN=ISP.SISPMACS,DISP=SHR
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
//SYSPRINT DD  SYSOUT=*
//    PEND
//*--------------------------------------------------------------------*
```

Figure 31. Sample IBMDEMO Project Definition (CLZTPDEF) (Part 1 of 3)

```
//LINK   PROC
//STEP2LNK EXEC PGM=IEWL,REGION=2048K,COND=(0,NE),
//             PARM='LIST,XREF,LET,RENT,REUS,CALL'
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DSN=IBMDEMO.PROJDEFS.OBJLIB,DISP=SHR
//SYSLMOD  DD  DSN=IBMDEMO.PROJDEFS.LOAD,DISP=SHR
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
//*-------------------------------------------------------------------*
//    PEND
//COMPILE   EXEC COMP
//SYSIN   DD *
         TITLE '*** PROJECT DEFINITION FOR PROJECT=IBMDEMO ***'
IBMDEMO  FLMABEG
*
*        ************************************************************
*        *  DEFINE THE AUTHORIZATON CODES                           *
*        ************************************************************
GRPDEV   FLMAGRP AC=(DEV)
*
*        ************************************************************
*        *  DEFINE THE TYPES                                        *
*        ************************************************************
COBOL    FLMTYPE ,                    HOST COBOL CODE
DOC      FLMTYPE ,                    WORD FOR WINDOWS DOCUMENTATION
GRAPHICS FLMTYPE ,                    GRAPHICS FILES (JPG, GIF)
HTML     FLMTYPE ,                    HTML AND JAVASCRIPT
JAR      FLMTYPE ,                    JAVA JAR
JAVA     FLMTYPE ,                    JAVA SOURCE
JAVACLAS FLMTYPE ,                    JAVA CLASSES
JAVALIST FLMTYPE ,                    JAVA LISTING
PACKAGES FLMTYPE ,                    PACKAGE ARCHHL
*
*        ************************************************************
*        *  DEFINE THE GROUPS                                       *
*        ************************************************************
DEV      FLMGROUP AC=(DEV),KEY=Y,PROMOTE=QA      DEVELOPMENT
QA       FLMGROUP AC=(DEV),KEY=Y,PROMOTE=REL     QUALITY ASSURANCE
REL      FLMGROUP AC=(DEV),KEY=Y                 PRODUCTION
*
*********************************************************************
*               PROJECT PROJDEFSS
*********************************************************************
        FLMCNTRL ACCT=IBMDEMO.PROJDEFS.ACCT,                          X
              VERS=IBMDEMO.PROJDEFS.VERSION,                          X
              XREF=IBMDEMO.PROJDEFS.XREF,                             X
              LIBID=SCLM
*
```

Figure 31. Sample IBMDEMO Project Definition (CLZTPDEF) (Part 2 of 3)

```
*********************************************************************
*               VERSIONING AND AUDITIBILITY                         *
*********************************************************************
        FLMATVER GROUP=DEV,TYPE=COBOL,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=DOC,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=GRAPHICS,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=HTML,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAR,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAVA,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAVACLAS,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAVALIST,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=PACKAGES,VERSION=YES
*
        FLMATVER GROUP=QA,TYPE=COBOL,VERSION=YES
        FLMATVER GROUP=QA,TYPE=DOC,VERSION=YES
        FLMATVER GROUP=QA,TYPE=GRAPHICS,VERSION=YES
        FLMATVER GROUP=QA,TYPE=HTML,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAR,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAVA,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAVACLAS,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAVALIST,VERSION=YES
        FLMATVER GROUP=QA,TYPE=PACKAGES,VERSION=YES
*
        FLMATVER GROUP=REL,TYPE=COBOL,VERSION=YES
        FLMATVER GROUP=REL,TYPE=DOC,VERSION=YES
        FLMATVER GROUP=REL,TYPE=GRAPHICS,VERSION=YES
        FLMATVER GROUP=REL,TYPE=HTML,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAR,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAVA,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAVACLAS,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAVALIST,VERSION=YES
        FLMATVER GROUP=REL,TYPE=PACKAGES,VERSION=YES
*
*********************************************************************
*               LANGUAGE DEFINITION TABLES                          *
*********************************************************************
*             TRANSLATOR           LANGUAGE
*       COPY  FLM@ARCD          -- ARCHDEF                   --
*       COPY  FLM@COB2          -- COBOL                     --
        COPY  CLZTJAR           -- JAR                       --
        COPY  CLZTJAVA          -- JAVA                      --
        COPY  CLZTJBIZ          -- E-BUSINESS OBJECTS        --
*
*********************************************************************
        FLMAEND
/*
//SYSLIN DD DSN=IBMDEMO.PROJDEFS.OBJLIB(IBMDEMO),DISP=SHR
//LINK   EXEC LINK
//SYSLIN DD *
 INCLUDE SYSLIB(IBMDEMO)
 NAME IBMDEMO(R)
/*
```

Figure 31. Sample IBMDEMO Project Definition (CLZTPDEF) (Part 3 of 3)

Note: If you want to make use of both the S-JDK USS build and the S-JDK FTP build, you can create separate translators for each build and use different language types in each. For example: instead of using LANG=JAVA in the CLZTJAVA translator, set up two translators, one of which specifies LANG=JAVAUSS and the other specifies LANG=JAVAFTP. The Language Definition Table in the Project definition must then include both of these translators.

Optionally, if you are building a new isolated project for the S-JDK system, you need to run two additional jobs: CLZTALIB and CLZTAVSM. The CLZTALIB JCL stream allocates all of the group libraries and CLZTAVSM allocates the SCLM VSAM files.

Step 4a: Allocate new S-JDK type data set

Allocate SCLM type files — CLZTALIB

Regardless of whether you build the types into an existing project or as a stand alone project, each type needs a set of libraries. The following JCL stream allocates these required libraries.

```
//* (JOBCARD)
//*--------------------------------------------------------------------*
//*  NAME:     CLZTALIB                                                *
//*  PURPOSE:  DELETE AND ALLOCATE THE S-JDK BASE AND VERSION FILES.   *
//*--------------------------------------------------------------------*
//*  TO USE THIS JCL YOU MUST:                                         *
//*         1) INSERT A VALID JOB CARD.                                *
//*         2) REVIEW THE DATASET NAMES - THEY ARE DELIVERED           *
//*            MATCHING THE S-JDK DEFAULT VALUES.                      *
//*         3) MODIFY THE DATASET NAMES TO MATCH ACTUAL CHOSEN S-JDK   *
//*            TYPES.                                                  *
//*         4) CHANGE THE UNIT=DUNIT TO THE APPROPRIATE UNIT FOR       *
//*            PERMANENT FILES.                                        *
//*--------------------------------------------------------------------*
//*  DELETE ALL S-JDK TYPE FILES                                       *
//*--------------------------------------------------------------------*
//DELETE  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
 DELETE IBMDEMO.DEV.COBOL
 DELETE IBMDEMO.DEV.DOC
 DELETE IBMDEMO.DEV.GRAPHICS
 DELETE IBMDEMO.DEV.HTML
 DELETE IBMDEMO.DEV.JAR
 DELETE IBMDEMO.DEV.JAVA
 DELETE IBMDEMO.DEV.JAVACLAS
 DELETE IBMDEMO.DEV.JAVALIST
 DELETE IBMDEMO.DEV.JCL
 DELETE IBMDEMO.DEV.PACKAGES
 DELETE IBMDEMO.QA.COBOL
 DELETE IBMDEMO.QA.DOC
 DELETE IBMDEMO.QA.GRAPHICS
 DELETE IBMDEMO.QA.HTML
 DELETE IBMDEMO.QA.JAR
 DELETE IBMDEMO.QA.JAVA
 DELETE IBMDEMO.QA.JAVACLAS
 DELETE IBMDEMO.QA.JAVALIST
 DELETE IBMDEMO.QA.JCL
 DELETE IBMDEMO.QA.PACKAGES
 DELETE IBMDEMO.REL.COBOL
 DELETE IBMDEMO.REL.DOC
 DELETE IBMDEMO.REL.GRAPHICS
 DELETE IBMDEMO.REL.HTML
 DELETE IBMDEMO.REL.JAR
 DELETE IBMDEMO.REL.JAVA
 DELETE IBMDEMO.REL.JAVACLAS
 DELETE IBMDEMO.REL.JAVALIST
 DELETE IBMDEMO.REL.JCL
 DELETE IBMDEMO.REL.PACKAGES
 DELETE IBMDEMO.DEV.COBOL.VERSION
 DELETE IBMDEMO.DEV.DOC.VERSION
 DELETE IBMDEMO.DEV.GRAPHICS.VERSION
 DELETE IBMDEMO.DEV.HTML.VERSION
 DELETE IBMDEMO.DEV.JAR.VERSION
 DELETE IBMDEMO.DEV.JAVA.VERSION
 DELETE IBMDEMO.DEV.JAVACLAS.VERSION
 DELETE IBMDEMO.DEV.JAVALIST.VERSION
 DELETE IBMDEMO.DEV.JCL.VERSION
 DELETE IBMDEMO.DEV.PACKAGES.VERSION
 DELETE IBMDEMO.QA.COBOL.VERSION
 DELETE IBMDEMO.QA.DOC.VERSION
```

Figure 32. CLZTALIB SCLM Type Data Set Allocations (Part 1 of 3)

```
 DELETE IBMDEMO.QA.GRAPHICS.VERSION
 DELETE IBMDEMO.QA.HTML.VERSION
 DELETE IBMDEMO.QA.JAR.VERSION
 DELETE IBMDEMO.QA.JAVA.VERSION
 DELETE IBMDEMO.QA.JAVACLAS.VERSION
 DELETE IBMDEMO.QA.JAVALIST.VERSION
 DELETE IBMDEMO.QA.JCL.VERSION
 DELETE IBMDEMO.QA.PACKAGES.VERSION
 DELETE IBMDEMO.REL.COBOL.VERSION
 DELETE IBMDEMO.REL.DOC.VERSION
 DELETE IBMDEMO.REL.GRAPHICS.VERSION
 DELETE IBMDEMO.REL.HTML.VERSION
 DELETE IBMDEMO.REL.JAR.VERSION
 DELETE IBMDEMO.REL.JAVA.VERSION
 DELETE IBMDEMO.REL.JAVACLAS.VERSION
 DELETE IBMDEMO.REL.JAVALIST.VERSION
 DELETE IBMDEMO.REL.JCL.VERSION
 DELETE IBMDEMO.REL.PACKAGES.VERSION
//*-------------------------------------------------------------------*
//*  PROC TO ALLOCATE BASE FILES                                      *
//*-------------------------------------------------------------------*
//ALLOC     PROC FILE=
//STEP1   EXEC PGM=IEFBR14
//DD01 DD  DSN=&FILE,
//     DISP=(NEW,CATLG,DELETE),
//     UNIT=DUNIT,
//     SPACE=(CYL,(2,10,100)),
//     DCB=(RECFM=VB,LRECL=256,BLKSIZE=0)
//   PEND
//*-------------------------------------------------------------------*
//*  PROC TO ALLOCATE VERSION FILES                                   *
//*-------------------------------------------------------------------*
//VALLOC    PROC FILE=
//STEP2   EXEC PGM=IEFBR14
//DD01 DD  DSN=&FILE,
//     DISP=(NEW,CATLG,DELETE),
//     UNIT=DUNIT,
//     SPACE=(CYL,(2,10,100)),
//     DCB=(RECFM=VB,LRECL=350,BLKSIZE=0)
//   PEND
//*-------------------------------------------------------------------*
//*  NOW DO THE ALLOCATES                                             *
//*-------------------------------------------------------------------*
//FILE01 EXEC ALLOC,FILE=IBMDEMO.DEV.COBOL
//FILE02 EXEC ALLOC,FILE=IBMDEMO.DEV.DOC
//FILE03 EXEC ALLOC,FILE=IBMDEMO.DEV.GRAPHICS
//FILE04 EXEC ALLOC,FILE=IBMDEMO.DEV.HTML
//FILE05 EXEC ALLOC,FILE=IBMDEMO.DEV.JAR
//FILE06 EXEC ALLOC,FILE=IBMDEMO.DEV.JAVA
//FILE07 EXEC ALLOC,FILE=IBMDEMO.DEV.JAVACLAS
//FILE08 EXEC ALLOC,FILE=IBMDEMO.DEV.JAVALIST
//FILE09 EXEC ALLOC,FILE=IBMDEMO.DEV.JCL
//FILE10 EXEC ALLOC,FILE=IBMDEMO.DEV.PACKAGES
//FILE11 EXEC ALLOC,FILE=IBMDEMO.QA.COBOL
//FILE12 EXEC ALLOC,FILE=IBMDEMO.QA.DOC
//FILE13 EXEC ALLOC,FILE=IBMDEMO.QA.GRAPHICS
//FILE14 EXEC ALLOC,FILE=IBMDEMO.QA.HTML
//FILE15 EXEC ALLOC,FILE=IBMDEMO.QA.JAR
//FILE16 EXEC ALLOC,FILE=IBMDEMO.QA.JAVA
//FILE17 EXEC ALLOC,FILE=IBMDEMO.QA.JAVACLAS
//FILE18 EXEC ALLOC,FILE=IBMDEMO.QA.JAVALIST
```

Figure 32. CLZTALIB SCLM Type Data Set Allocations (Part 2 of 3)

```
//FILE19 EXEC ALLOC,FILE=IBMDEMO.QA.JCL
//FILE20 EXEC ALLOC,FILE=IBMDEMO.QA.PACKAGES
//FILE21 EXEC ALLOC,FILE=IBMDEMO.REL.COBOL
//FILE22 EXEC ALLOC,FILE=IBMDEMO.REL.DOC
//FILE23 EXEC ALLOC,FILE=IBMDEMO.REL.GRAPHICS
//FILE24 EXEC ALLOC,FILE=IBMDEMO.REL.HTML
//FILE25 EXEC ALLOC,FILE=IBMDEMO.REL.JAR
//FILE26 EXEC ALLOC,FILE=IBMDEMO.REL.JAVA
//FILE27 EXEC ALLOC,FILE=IBMDEMO.REL.JAVACLAS
//FILE28 EXEC ALLOC,FILE=IBMDEMO.REL.JAVALIST
//FILE29 EXEC ALLOC,FILE=IBMDEMO.REL.JCL
//FILE30 EXEC ALLOC,FILE=IBMDEMO.REL.PACKAGES
//*
//FILE31 EXEC VALLOC,FILE=IBMDEMO.DEV.COBOL.VERSION
//FILE32 EXEC VALLOC,FILE=IBMDEMO.DEV.DOC.VERSION
//FILE33 EXEC VALLOC,FILE=IBMDEMO.DEV.GRAPHICS.VERSION
//FILE34 EXEC VALLOC,FILE=IBMDEMO.DEV.HTML.VERSION
//FILE35 EXEC VALLOC,FILE=IBMDEMO.DEV.JAR.VERSION
//FILE36 EXEC VALLOC,FILE=IBMDEMO.DEV.JAVA.VERSION
//FILE37 EXEC VALLOC,FILE=IBMDEMO.DEV.JAVACLAS.VERSION
//FILE38 EXEC VALLOC,FILE=IBMDEMO.DEV.JAVALIST.VERSION
//FILE39 EXEC VALLOC,FILE=IBMDEMO.DEV.JCL.VERSION
//FILE40 EXEC VALLOC,FILE=IBMDEMO.DEV.PACKAGES.VERSION
//FILE41 EXEC VALLOC,FILE=IBMDEMO.QA.COBOL.VERSION
//FILE42 EXEC VALLOC,FILE=IBMDEMO.QA.DOC.VERSION
//FILE43 EXEC VALLOC,FILE=IBMDEMO.QA.GRAPHICS.VERSION
//FILE44 EXEC VALLOC,FILE=IBMDEMO.QA.HTML.VERSION
//FILE45 EXEC VALLOC,FILE=IBMDEMO.QA.JAR.VERSION
//FILE46 EXEC VALLOC,FILE=IBMDEMO.QA.JAVA.VERSION
//FILE47 EXEC VALLOC,FILE=IBMDEMO.QA.JAVACLAS.VERSION
//FILE48 EXEC VALLOC,FILE=IBMDEMO.QA.JAVALIST.VERSION
//FILE49 EXEC VALLOC,FILE=IBMDEMO.QA.JCL.VERSION
//FILE50 EXEC VALLOC,FILE=IBMDEMO.QA.PACKAGES.VERSION
//FILE51 EXEC VALLOC,FILE=IBMDEMO.REL.COBOL.VERSION
//FILE52 EXEC VALLOC,FILE=IBMDEMO.REL.DOC.VERSION
//FILE53 EXEC VALLOC,FILE=IBMDEMO.REL.GRAPHICS.VERSION
//FILE54 EXEC VALLOC,FILE=IBMDEMO.REL.HTML.VERSION
//FILE55 EXEC VALLOC,FILE=IBMDEMO.REL.JAR.VERSION
//FILE56 EXEC VALLOC,FILE=IBMDEMO.REL.JAVA.VERSION
//FILE57 EXEC VALLOC,FILE=IBMDEMO.REL.JAVACLAS.VERSION
//FILE58 EXEC VALLOC,FILE=IBMDEMO.REL.JAVALIST.VERSION
//FILE59 EXEC VALLOC,FILE=IBMDEMO.REL.JCL.VERSION
//FILE60 EXEC VALLOC,FILE=IBMDEMO.REL.PACKAGES.VERSION
```

Figure 32. CLZTALIB SCLM Type Data Set Allocations (Part 3 of 3)

Step 4b (optional): Allocate new project VSAM files

Allocate project VSAM files — CLZTAVSM

If you are building an isolated, stand alone Project for the S-JDK types, there are three VSAM files that need to be allocated. The following JCL stream, found in the CLZ.SCLZJCL library, allocates these required libraries.

```
//* (JOBCARD)
//*-------------------------------------------------------------------*
//*  NAME:     CLZTAVSM                                               *
//*  PURPOSE:  DELETE AND ALLOCATE THE S-JDK PROJECT VSAM FILES.      *
//*-------------------------------------------------------------------*
//*  TO USE THIS JCL YOU MUST:                                        *
//*         1) INSERT A VALID JOB CARD.                               *
//*         2) REVIEW THE DATASET NAMES - THEY ARE DELIVERED          *
//*            MATCHING THE S-JDK DEFAULT VALUES.                     *
//*         3) MODIFY THE DATASET NAMES TO MATCH ACTUAL CHOSEN S-JDK  *
//*            TYPES.                                                 *
//*         4) CHANGE THE UNIT=TDISK TO THE APPROPRIATE UNIT FOR      *
//*            TEMPORARY FILES.                                       *
//*         5) CHANGE THE "DVOLSER" VARIABLE TO THE APPROPRIATE       *
//*            VOLUME FOR VSAM FILES.                                 *
//*********************************************************************
//*  IBMDEMO.PROJDEFS.JCLLIB(ALLOVSAM)
//*********************************************************************
//ACCT1   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     DELETE IBMDEMO.PROJDEFS.ACCT
     DEFINE CLUSTER +
        (NAME('IBMDEMO.PROJDEFS.ACCT') +
         CYLINDERS(1 1) +
         VOLUMES(DVOLSER) +
         KEYS(26 0) +
         RECORDSIZE(264 32000) +
         SHAREOPTIONS(4,3) +
         SPEED +
         SPANNED +
         UNIQUE) +
         INDEX(NAME('IBMDEMO.PROJDEFS.ACCT.I') -
         ) +
         DATA(NAME('IBMDEMO.PROJDEFS.ACCT.D') -
         CISZ(2048) +
         FREESPACE(50 50) +
         )
/*
//*********************************************************************
//*
//*  INITIALIZE THE ACCOUNTING FILE
//*
//*********************************************************************
//ACCT2    EXEC PGM=IDCAMS
//INPUT  DD *
                        SCLM ACCOUNTING FILE INITIALIZATION RECORD
/*
//OUTPUT DD DSN=IBMDEMO.PROJDEFS.ACCT,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
```

Figure 33. CLZTAVSM — Allocate Project VSAM Files (Part 1 of 3)

```
//*********************************************************************
//*  A JOB STEP IS THEN EXECUTED TO INITIALIZE THE FILE.
//*********************************************************************
//VERS1   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     DELETE IBMDEMO.PROJDEFS.VERSION
     DEFINE CLUSTER +
        (NAME('IBMDEMO.PROJDEFS.VERSION') +
         CYLINDERS(1 1) +
         VOLUMES(DVOLSER) +
         KEYS(40 0) +
         RECORDSIZE(264 32000) +
         SHAREOPTIONS(4,3) +
         SPEED +
         SPANNED +
         UNIQUE) +
         INDEX(NAME('IBMDEMO.PROJDEFS.VERSION.I') -
         ) +
         DATA(NAME('IBMDEMO.PROJDEFS.VERSION.D') -
         CISZ(2048) +
         FREESPACE(50 50) +
         )
/*
/*
//*********************************************************************
//*  INITIALIZE THE AUDIT CONTROL FILE
//*********************************************************************
//VERS2    EXEC PGM=IDCAMS
//INPUT  DD *
                        SCLM AUDIT CONTROL FILE INITIALIZATION RECORD
/*
//OUTPUT DD DSN=IBMDEMO.PROJDEFS.VERSION,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*********************************************************************
//*  A JOB STEP IS THEN EXECUTED TO INITIALIZE THE FILE.
//*
//*********************************************************************
//XREF0   EXEC PGM=BZZFPARM,
//  PARM='NEW,SCLM CROSS REFERENCE INITIALIZATION RECORD'
//STEPLIB DD DSN=IBMDEMO.PRODUCT.LOADLIB,DISP=SHR
//CIGPOUT DD DSN=&&PARMFILE,DISP=(NEW,PASS),
//   UNIT=TDISK,DCB=(LRECL=128,BLKSIZE=12800,RECFM=FB),
//   SPACE=(TRK,1)
//CIGLOG  DD SYSOUT=*
//* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
//XREF1   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     DELETE IBMDEMO.PROJDEFS.XREF
     DEFINE CLUSTER +
        (NAME('IBMDEMO.PROJDEFS.XREF') +
         CYLINDERS(2 1) +
         VOLUMES(DVOLSER) +
         KEYS(128 0) +
         RECORDSIZE(264 32000) +
         SHAREOPTIONS(4,3) +
```

Figure 33. CLZTAVSM — Allocate Project VSAM Files (Part 2 of 3)

```
 SPEED +
        SPANNED +
        UNIQUE) +
        INDEX(NAME('IBMDEMO.PROJDEFS.XREF.I') -
        ) +
        DATA(NAME('IBMDEMO.PROJDEFS.XREF.D') -
        CISZ(2048) +
        FREESPACE(50 50) +
        )
/*
//*********************************************************************
//*  INITIALIZE THE CROSS-REFERENCE FILE
//*********************************************************************
//XREF2    EXEC PGM=IDCAMS
//INPUT  DD DSN=&&PARMFILE,DISP=(OLD,DELETE)
//OUTPUT DD DSN=IBMDEMO.PROJDEFS.XREF,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*
```

Figure 33. CLZTAVSM — Allocate Project VSAM Files (Part 3 of 3)

Step 5: Define S-JDK types to Cloud 9 SLR

Thhis step defines the S-JDK types to your Cloud 9 SLR file. The following example shows the default S-JDK definitions required to define the S-JDK types to the Cloud 9 SLR. For information about the syntax of the SLR utility statements, see Appendix B, “Suite Long Name Registry,” on page 213.

Modify and submit CLZC9J06

1. Using ISPF EDIT, access member CLZC9J06 (CLZ.SCLZJCL library). This JCL should already have been modified during the base installation and IVP process.
2. Update the member, including the following SLR definitions (in BOLD).
3. Submit the job.

Note: This job should end with COND CODE=0.

```
//* (JOBCARD)
//* ------------------------------------------------------------------*
//* CLOUD 9 JAVA/S-JDK COMPONENTS.                                     *
//* ------------------------------------------------------------------*
//* ------------------------------------------------------------------*
//*  NAME:    CLZC9J06                                                 *
//*  PURPOSE: DEFINE S-JDK TYPES TO THE SLR DATABASE.                  *
//* ------------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                        *
//*         1) INSERT A VALID JOB CARD WITH VALID CLASS AND REGION=0M *
//*         2) MAKE SURE THAT THE STEPLIB POINTS TO YOUR CLOUD9       *
//*            AUTHORIZED DATASET THAT CONTAINS THE CIGINI.           *
//*         3) MODIFY THE TYPE NAMES IF YOU HAVE CHANGED THE DEFAULT  *
//*            SCLM TYPES.                                            *
//* ------------------------------------------------------------------ *
//STEP1    EXEC PGM=CZLSLR
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR
//CIGIN    DD *
ADD NAME RULE FOR SCLM TYPE DOC      CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE GRAPHICS CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE HTML     CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE JAVA     CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE JAR      CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE JAVACLAS CASE SENSITIVE.
//CIGLOG   DD SYSOUT=*
```

Figure 34. CLZC9J06 — Define S-JDK Types to Cloud 9 SLR

Step 6: Run CLZTAUNX to Build S-JDK USS directories

This step defines the S-JDK UNIX directories. The member is shown with default values.

Modify and submit CLZTAUNX

1. Using ISPF EDIT, access member CLZTAUNX (CLZ.SCLZJCL library).
2. Issue the CAPS OFF command to ensure case sensitivity.
3. Copy your job card values to the top of the member. Substitute your site-specific values (identified on the Installation Worksheet) for those values in bold in Figure 35 on page 94.
4. Submit the job.

Note: This job should end with COND CODE=0.

```
//* (JOBCARD)
//*--------------------------------------------------------------------*
//*  NAME:     CLZTAUNX                                                *
//*  PURPOSE:  UNIX ALLOCATION OF S-JDK DIRECTORIES AND PERMISSIONS.   *
//*--------------------------------------------------------------------*
//*  TO USE THIS JCL YOU MUST:                                         *
//*         1) INSERT A VALID JOB CARD.                                *
//*         2) REVIEW THE DIRECTORY NAMES - THEY ARE DELIVERED         *
//*            MATCHING THE S-JDK DEFAULT VALUES.                      *
//*         3) MODIFY THE DIRECTORY NAME AS PER THE SCLM/USS           *
//*            WORKSHEET.                                              *
//*--------------------------------------------------------------------*
//*  UNIX CLEANUP                                                      *
//*--------------------------------------------------------------------*
//UNIX    EXEC PGM=IKJEFT01
//SYSPROC  DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSIN DD  *

   /* Remove all files in ibmdemo */
   oshell rm -r /u/ibmdemo/dev
   oshell rm -r /u/ibmdemo/qa
   oshell rm -r /u/ibmdemo/rel

   /* Remove all directories in ibmdemo-dev */
   oshell rmdir -p /u/ibmdemo/dev/classes
   oshell rmdir -p /u/ibmdemo/dev/graphics
   oshell rmdir -p /u/ibmdemo/dev/html
   oshell rmdir -p /u/ibmdemo/dev/jar
   oshell rmdir -p /u/ibmdemo/dev/listings

   /* Remove all directories in ibmdemo-qa  */
   oshell rmdir -p /u/ibmdemo/qa/classes
   oshell rmdir -p /u/ibmdemo/qa/graphics
   oshell rmdir -p /u/ibmdemo/qa/html
   oshell rmdir -p /u/ibmdemo/qa/jar
   oshell rmdir -p /u/ibmdemo/qa/listings
```

Figure 35. CLZTAUNX — Create USS S-JDK Directories (Part 1 of 2)

```
/* Remove all directories in ibmdemo-rel */
  oshell rmdir -p /u/ibmdemo/rel/classes
  oshell rmdir -p /u/ibmdemo/rel/graphics
  oshell rmdir -p /u/ibmdemo/rel/html
  oshell rmdir -p /u/ibmdemo/rel/jar
  oshell rmdir -p /u/ibmdemo/rel/listings

  /* Make all directories in ibmdemo-dev */
  oshell mkdir -p /u/ibmdemo/dev/classes
  oshell mkdir -p /u/ibmdemo/dev/graphics
  oshell mkdir -p /u/ibmdemo/dev/html
  oshell mkdir -p /u/ibmdemo/dev/jar
  oshell mkdir -p /u/ibmdemo/dev/listings

  /* Make all directories in ibmdemo-qa  */
  oshell mkdir -p /u/ibmdemo/qa/classes
  oshell mkdir -p /u/ibmdemo/qa/graphics
  oshell mkdir -p /u/ibmdemo/qa/html
  oshell mkdir -p /u/ibmdemo/qa/jar
  oshell mkdir -p /u/ibmdemo/qa/listings

  /* Make all directories in ibmdemo-rel */
  oshell mkdir -p /u/ibmdemo/rel/classes
  oshell mkdir -p /u/ibmdemo/rel/graphics
  oshell mkdir -p /u/ibmdemo/rel/html
  oshell mkdir -p /u/ibmdemo/rel/jar
  oshell mkdir -p /u/ibmdemo/rel/listings

  /* Set permissions for ibmdemo projects */
  oshell chmod -R 777 /u/ibmdemo/dev
  oshell chmod -R 777 /u/ibmdemo/qa
  oshell chmod -R 777 /u/ibmdemo/rel

  /* copy sample files to target location */
 oput 'CLZ.SCLZHTML(CLZHCLCK)' '/u/ibmdemo/dev/clock.html'
 oput 'CLZ.SCLZHTML(CLZJCLCK)' '/u/ibmdemo/dev/Clock2.java'

 oshell chmod 777 '/u/ibmdemo/dev/clock.html'
 oshell chmod 777 '/u/ibmdemo/dev/Clock2.java'

//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//OSHELL   DD SYSOUT=*
```

Figure 35. CLZTAUNX — Create USS S-JDK Directories (Part 2 of 2)

Step 7: Review CLZJIBM UNIX shell — delete processing

Understanding SCLM delete processing

To perform delete processing, the user must run the delete utility provided for deletion of JAVA/USS compiled and copied objects. This utility is resident in the CLZJIBM JCL shell that comes with the Cloud 9 product.

The following figure contains the CLZJIBM JCL shell as delivered with the product. Note, at the end of the member, there is a Delete Action step that invokes one of the S-JDK REXX utilities. Review this step for consistency with other customizations that have been performed against the translators and control files.

```
)DOT
%JOBCARD%
)ENDDOT
//* ------------------------------------------------------------------ *
//* NAME:     CLZJIBM                                                  *
//* PURPOSE:  CLOUD 9 FOR SCLM.                                        *
//*           SCLM BATCH SKELETON.                                     *
//* ------------------------------------------------------------------ *
//*                                                                    *
//* REQUIRED JCL MODIFICATION:                                         *
//*  1) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.        *
//*      - ISPFQUAL                                                    *
//*      - TDISK                                                       *
//*  2) TARGET LIBRARIES CLZ.SCLZLOAD AND CLZ.SCLZCGI        /* C1 */ *
//*     SHOULD BE CHANGED IF THE INSTALLED HIGH LEVEL        /* C1 */ *
//*     QUALIFIER IS NOT 'CLZ.'                              /* C1 */ *
//*                                                                    *
//*  NOTE: BREEZE USERS MUST MODIFY THIS MODULE PER EMBEDDED           *
//*        INSTRUCTION. BREEZE DATASET NAMES MAY ALSO NEED TO BE       *
//*        MODIFIED.                                                   *
//*                                                                    *
//* 22OCT2001 OW51810 - CHANGES MARKED AS /* C1 */                     *
//* Z020402A                                                           *
//*                                                                    *
//*--------------------------------------------------------------------*
//*  RESIDES IN HTTP SERVER AT:                           /* C1 */ *
//*  /ROOTDIR/CLOUD9/JCL/CLZJIBM                                       *
//*--------------------------------------------------------------------*
)IF ACTION=OCOPY
//COPY   EXEC PGM=IKJEFT01
)DOT
%COPYFILES%
)ENDDOT
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
)DOT
%OCOPYSYNTAX%
)ENDDOT
/*
)ENDIF
)IF ACTION=IEBCOPY
//COPY     EXEC PGM=IEBCOPY
)DOT
%COPYFILES%
)ENDDOT
//SYSIN   DD *
)DOT
%OCOPYSYNTAX%
)ENDDOT
//SYSPRINT  DD  SYSOUT=*
)ENDIF
```

Figure 36. CLZJIBM UNIX JCL Shell (Part 1 of 4)

```
//*-------------------------------------------------------------------
//GENER  EXEC PGM=IEBGENER
//SYSUT1   DD *
)DOT
%SCLMSYNTAX%
)ENDDOT
//SYSUT2   DD DSN=&&CLIST(TEMPNAME),UNIT=TDISK,
//            SPACE=(TRK,(10,10,2),RLSE),
//            DISP=(NEW,PASS),DCB=(LRECL=80,
//            BLKSIZE=1600,DSORG=PO,RECFM=FB)
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
)IF ACTION=DELETE
//DGRPTS   DD DSN=&&DELLIST,DISP=(NEW,PASS),                DELETE
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
)IF ACTION=BUILD
//COPYBULD EXEC PGM=CLZTFILE                                JAVA
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR                        /* C1 */
//SYSIN    DD *                                             JAVA
)DOT
%SCLMSYNTAX%
)ENDDOT
//SYSOUT   DD DSN=&&BSYNTAX,DISP=(NEW,PASS),                JAVA
//            SPACE=(TRK,(10,10),RLSE),UNIT=TDISK,          JAVA
//            DCB=(LRECL=80,BLKSIZE=0,DSORG=PS,RECFM=FB)    JAVA
)ENDIF
//*****************************************************************
//* BREEZE USERS:  UNCOMMENT LINES WITH "BREEZE USERS"
//*****************************************************************
//TSO    EXEC PGM=IKJEFT01,REGION=4096K,TIME=1439,DYNAMNBR=200
//STEPLIB  DD DSN=CLZ.SCLZLOAD,DISP=SHR
//*        DD DSN=BZZ.SBZZLOAD,DISP=SHR            BREEZE USERS
//SYSTSIN  DD *
  ISPSTART CMD(%TEMPNAME)
//SYSTSPRT DD SYSOUT=(*)
//SYSPROC  DD DSN=&&CLIST,DISP=(OLD,DELETE)
//*        DD DSN=BZZ.SBZZCLIB,DISP=SHR            BREEZE USERS
//*****************************************************************
```

Figure 36. CLZJIBM UNIX JCL Shell (Part 2 of 4)

```
//* ISPF LIBRARIES
//ISPMLIB  DD DSN=ISPFQUAL.SISPMENU,DISP=SHR
//*        DD DSN=BZZ.SBZZMENU,DISP=SHR            BREEZE USERS
//ISPSLIB  DD DSN=ISPFQUAL.SISPSENU,DISP=SHR
//         DD DSN=ISPFQUAL.SISPSLIB,DISP=SHR
//*        DD DSN=BZZ.SBZZSENU,DISP=SHR            BREEZE USERS
//ISPPLIB  DD DSN=ISPFQUAL.SISPPENU,DISP=SHR
//*        DD DSN=BZZ.SBZZPENU,DISP=SHR            BREEZE USERS
//ISPTLIB  DD UNIT=VIO,DISP=(NEW,PASS),SPACE=(CYL,(1,1,5)),
//            DCB=(LRECL=80,BLKSIZE=19040,DSORG=PO,RECFM=FB)
//         DD DSN=ISPFQUAL.SISPTENU,DISP=SHR
//ISPTABL  DD UNIT=VIO,DISP=(NEW,PASS),SPACE=(CYL,(1,1,5)),
//            DCB=(LRECL=80,BLKSIZE=19040,DSORG=PO,RECFM=FB)
//ISPPROF  DD UNIT=VIO,DISP=(NEW,PASS),SPACE=(CYL,(1,1,5)),
//            DCB=(LRECL=80,BLKSIZE=19040,DSORG=PO,RECFM=FB)
//ISPLOG   DD SYSOUT=*,
//            DCB=(LRECL=120,BLKSIZE=2400,DSORG=PS,RECFM=FB)
//ISPCTL1  DD DISP=NEW,UNIT=VIO,SPACE=(CYL,(1,1)),
//            DCB=(LRECL=80,BLKSIZE=800,RECFM=FB)   TEMPORARY FILE
//ZFLMDD   DD *
    ZFLMNLST=FLMNLENU  ZFLMTRMT=ISR3278   ZDATEF=YY/MM/DD
/*
//*CIGLOG  DD SYSOUT=*                             BREEZE USERS
//*CIGLOG0 DD SYSOUT=*                             BREEZE USERS
//*CIGLOG1 DD DSN=&&CIGLOG1,DISP=(NEW,DELETE),     BREEZE USERS
//*           UNIT=TDISK,SPACE=(CYL,(1,1)),        BREEZE USERS/* C1 */
//*           DCB=(LRECL=132,BLKSIZE=0,RECFM=FB)   BREEZE USERS
//*CIGLOG2 DD DSN=&&CIGLOG2,DISP=(NEW,DELETE),     BREEZE USERS
//*           UNIT=TDISK,SPACE=(CYL,(1,1)),        BREEZE USERS/* C1 */
//*           DCB=(LRECL=132,BLKSIZE=0,RECFM=FB)   BREEZE USERS
//*CIGLOG3 DD DSN=&&CIGLOG3,DISP=(NEW,DELETE),     BREEZE USERS
//*           UNIT=TDISK,SPACE=(CYL,(1,1)),        BREEZE USERS/* C1 */
//*           DCB=(LRECL=132,BLKSIZE=0,RECFM=FB)   BREEZE USERS
//*****************************************************************
```

Figure 36. CLZJIBM UNIX JCL Shell (Part 3 of 4)

```
//* SCLM OUTPUT FILES
//*****************************************************************
//FLMMSGS  DD SYSOUT=(*)
)IF ACTION=BUILD
//BLDMSGS  DD SYSOUT=*,                                        BUILD
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//BLDREPT  DD SYSOUT=*,                                        BUILD
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FBA)
//BLDLIST  DD SYSOUT=*,                                        BUILD
//         DCB=(LRECL=259,BLKSIZE=3120,RECFM=VB)
//BLDEXIT  DD DSN=&&BLDEXIT,DISP=(NEW,DELETE),                 BUILD
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=160,BLKSIZE=3200,RECFM=FB)
//BSYNTAX  DD DSN=&&BSYNTAX,DISP=(OLD,PASS)                    JAVA
)ENDIF
)IF ACTION=PROMOTE
//PROMMSGS DD SYSOUT=*,                                        PROMOTE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=FB,DSORG=PS)
//PROMREPT DD SYSOUT=*,                                        PROMOTE
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB,DSORG=PS)
//PROMEXIT DD DSN=&&PROMEXIT,DISP=(NEW,DELETE),                PROMOTE
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=160,BLKSIZE=3200,RECFM=FB)
//COPYERR  DD SYSOUT=*,                                        PROMOTE
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
)ENDIF
)IF ACTION=MIGRATE
//U2LSTS   DD SYSOUT=*,                                        MIGRATE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//U2MSGS   DD SYSOUT=*,                                        MIGRATE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
)IF ACTION=DELETE
//DGLIST   DD SYSOUT=*,                                        DELETE
//         DCB=(LRECL=137,BLKSIZE=3120,RECFM=VBA)
//DGMSGS   DD SYSOUT=*,                                        DELETE
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//DGREPT   DD DSN=&&DELLIST,DISP=(MOD,PASS)
//DGEXIT   DD DSN=&&DELEXIT,DISP=(NEW,DELETE),                 DELETE
//         SPACE=(TRK,(5,10),RLSE),
//         DCB=(LRECL=160,BLKSIZE=3200,RECFM=FB)
)ENDIF
)IF ACTION=VERRECOV
//DBUMSGS  DD SYSOUT=*,                                        VERRECOV
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
//*-----------------------------------------------------------------------
)IF ACTION=DELETE
//DELMSGS EXEC PGM=IEBGENER
//SYSUT1   DD DSN=&&DELLIST,DISP=(OLD,PASS)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//*
//DELETE  EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=200
//STEPLIB  DD  DSN=CLZ.SCLZLOAD,DISP=SHR
//SYSTSIN  DD  *
  EX 'CLZ.SCLZCGI(CLZTRJDL)'
//SYSTSPRT DD  SYSOUT=*
//DELLIST  DD DSN=&&DELLIST,DISP=(OLD,PASS)
//LISTOUT  DD SYSOUT=*,
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//UNIXLOC  DD DSN=CLZ.SCLZCGI(CLZTULOC),DISP=SHR
)ENDIF
```

Figure 36. CLZJIBM UNIX JCL Shell (Part 4 of 4)

Checkpoint #3 for S-JDK for USS Installation

At this point all SCLM and Cloud 9 definitions should be complete.

Table 22. Checkpoint #3 for S-JDK for USS installation

Task	Completed?
Update your project definition with JDK Types and Translators (Optionally CLZTPDEF)	
Allocate New SCLM Type Libraries CLZTALIB	
Optionally allocate new SCLM project VSAM, only if creating a new project using CLZTAVSM	
Run CLZC9J06 to define S-JDK types to Cloud 9	
Run CLZTAUNX to define USS directories for S-JDK application	
Modified CLZJIBM Cloud 9 JCL shell	

Chapter 10. Perform Installation Verification Procedures

Step 8: Test the S-JDK translators

There are two files delivered with the S-JDK for translator verification. One is a JAVA file called CLZJCLCK and one is an HTML invocation file called CLZHCLCK. There are all delivered in the CLZ.SCLZHTML libraries. The CLZTAUNX JCL stream already copied and renamed the files to the JAVA application area, as shown in the following table.

Table 23. Translator verification files

Member	Already saved into USS as:
CLZJCLCK	/ibmdemo/dev/Clock2.java
CLZHCLCK	/ibmdemo/dev/clock.html

To verify that you have correctly installed and tailored the S-JDK translators:

1. Invoke Cloud 9.
2. List UNIX Files for the /u/ibmdemo/dev directories.
3. Select Clock2.java from list.
4. Migrate Clock2.java into Cloud 9.
   a. Add this with a type of JAVA and language of JAVA.
   b. You should see a short name generated.
5. List SCLM files for the JAVA type.
6. Build the Clock2.java to invoke the translators.
7. Review the expansion of the translator.
8. Review the population of the USS output life cycle and Java USS Output locations. If everything has worked correctly, the CLASSES directory contains:

```
/u/ibmdemo/dev/classes
   Type  Filename
 _ Dir   .
 _ Dir   ..
 _ File  Clock2.class
```

Figure 37. Directory Entry After Java Compile

Java listing example

Figure 38 on page 102 shows what you see in the Clock2.java listing file:

```
BROWSE -- /u/test/a/sysj/subj/listings/Clock2.java - Line 00000000 Col 001 080
 Command ===>                                                  Scroll ===> CSR
******************************** Top of Data ********************************
.parsed Clock2.java in 18438 ms.
.loaded /usr/lpp/java/J1.1/bin/../lib/classes.zip(java/applet/Applet.class) in 1
.loaded /usr/lpp/java/J1.1/bin/../lib/classes.zip(java/awt/Panel.class) in 210 m
.loaded /usr/lpp/java/J1.1/bin/../lib/classes.zip(java/awt/Container.class) in 1
.loaded /usr/lpp/java/J1.1/bin/../lib/classes.zip(java/awt/Component.class) in 2
.loaded /usr/lpp/java/J1.1/bin/../lib/classes.zip(java/awt/Component$NativeInLig
.loaded /usr/lpp/java/J1.1/bin/../lib/classes.zip(java/lang/Object.class) in 202
.loaded /usr/lpp/java/J1.1/bin/../lib/classes.zip(java/lang/Runnable.class) in 4
.checking class Clock2.
...more
```

Figure 38. Listing example from USS JAVA compile

Java SCLM Build Map example

Figure 39 shows an example of the component list for Clock2.java.

```
Command ===>                                                  Scroll ===> PAGE
******************************** Top of Data ********************************
                              Build Map Contents
                              ------------------

Keyword  Member                                Type     Last Time Modified Ver
-------- ------------------------------------ -------- ------------------ ---
SINC     CLO00001                             JAVA     01/05/05  20:09:00 1
LIST     CLO00001                             JAVALIST 01/05/07  15:00:21 26
OUT1     CLO00000                             JAVACLAS 01/05/07  15:00:21 13
****************************** Bottom of Data ******************************
```

Figure 39. Build Map list example

Step 9: Invoking the compiled Java

Copy the Clock2.class and the clock.html to a known location on a Web Server. For example, copy clock.html to the HTTP root directory and copy the Clock2.class to HTTP root directory/classes/ directory. You should be able to invoke the compiled Java code by entering the same IP address and port as Cloud 9, but instead of using Cloud9.htm, use the clock.html request.

In the location area of your browser, enter:

ip-address:port/clock.html

Checkpoint #4 for S-JDK for USS installation

At this point the following tasks should be complete.

Table 24. Checkpoint #4 for S-JDK for USS installation

Task	Completed?
Copied CLZJCLCK to a Cloud 9 or a USS location as Clock2.java	
Added Clock2.java as a JAVA element into SCLM using Cloud 9	
Reviewed translator output	
Reviewed population of USS Output directories	
Copied Clock2.class to the HTTP /rootdir/classes/ directory	
Invoked *clock.html* from a Web server location	

Part 3. SCLM-Java Development Kit for FTP Remote Build and Deploy

Chapter 11. S-JDK for FTP remote build and deploy installation overview

This part of the manual contains the installation procedure for the IBM Cloud 9 for SCLM for z/OS Remote Build and Deploy Java Development Kit feature. This is a feature which makes use of FTP to build or deploy Java code to another computer that is running an FTP server. The other computer can be a Windows or a Unix system.

Using certain FTP commands and the Java compiler installed on that box, Cloud 9 invokes a Java compile, creating class and listing outputs. These are then sent back to be stored in SCLM and can be deployed to other computers running FTP servers. Hereinafter, the following names are used in this manual:

- IBM Cloud 9 for SCLM for z/OS is called ***Cloud 9***
- IBM Cloud 9 for SCLM for z/OS Java Development Kit is called ***S-JDK***
- Remote Build and Deploy is called ***RBD***.

The steps in this part are organized into four major sections:

- Before you begin
- Customizing translators and translator control files
- Defining the S-JDK inventory, USS and Cloud 9 parts
- Performing Installation Verification Procedures (IVP)

READ THIS FIRST!

This is not an 'out of the box' solution. You should not use global editing on these files. You must thoroughly understand your FTP server and Java environment before attaching the SCLM translators.

The SCLM part of this solution is standard SCLM. There are translators, types, languages and control files. Some SCLM administrators might be unfamiliar with the JAVA and FTP Server side of the setup, therefore, you need to thoroughly review the system and software requirements for S-JDK (see Chapter 12, "Before you begin," on page 111), **paying particular attention to the FTP Server Prerequisites ("FTP server prerequisites" on page 111)**, before moving on to setting up the prototype translators.

Overview of the SCLM Remote Build and Deploy Java Development Kit

The purpose of the SCLM Remote Build and Deploy Java Development Kit (S-JDK RBD) is to provide the developer with the means to add e-business type objects, such as wordprocessing documents, spreadsheets, graphics files, HTML, XML and Java objects, to SCLM. This solution differs from the SCLM Java Development Kit for USS, described in Part 2, in that with this solution you can build your Java objects on a remote platform, such as an NT server, and then deploy them to other computers running FTP servers. In this part of the guide, you set up the translators and control files that control the e-business object management using File Transfer Protocol (FTP).

The three translators provided with this release, CLZTJAVA, CLZTJAR and CLZTJBIZ are used to manage and build Java Source, Jar make files and other

binary and text e-business objects. These are standard SCLM translators that use control files to drive the copying and compiling of the e-business objects.

At all times, the SCLM PDS's are the repository for all the code. The USS is used as an area in which to build, store and run e-business type applications such as HTML and Java. Source is copied there by the build process and outputs, such as Java class files, are copied back into SCLM using Cloud 9, on successful completion of a compile. This ensures that class files for a particular Java source file are all kept together in SCLM. The mapping control files that you tailor in this section show the SCLM life cycle name and the USS directory to which it maps. In this way, Cloud 9 can copy to and from the FTP Server directories during the build processes.

You can use the Cloud 9 SLR database (Short to Long name Registry) to store objects that have long file names, such as a Windows or UNIX file named ProjectOverview.doc, in SCLM. When you add a UNIX or PC file that has a long name to SCLM, Cloud 9 stores the long name in the SLR, along with a short name that it generates. This makes it possible to store members that conform to MVS naming conventions in SCLM PDS's, but maintain a record of their actual long file name in the SLR. Anytime you work with these members in the Cloud 9 lists, you see the long name but if you look at them from native SCLM, you see the generated short name.

To give an overview of what happens during a Java program compile, we list the steps here. To see how each of these steps is performed, check the Cloud 9 User Guide. This overview describes how the translators and control members fit into the process. You perform these steps as part of the IVP after tailoring has occurred.

1. Use Cloud 9 to migrate the Java Source into SCLM, giving it a language of JAVA. The language is related to the LANG= parameter in the required translator (CLZTJAVA).
2. In Cloud 9, issue a Build against the member. At this time Cloud 9 invokes the CLZTJAVA translator.
3. The CLZTJAVA translator initially goes to the SLR to get the Short name, so that it knows what the actual SCLM member is called.
4. It then uses the CLZTULOW to get the IP address, the encoded user ID/password and the Windows directory where it is going to copy the Java source (based on it's actual location within SCLM). When it does the copy, it copies from SCLM to the FTP-Server, using the short name in SCLM and the long name on the FTP Server.
5. The translator then calls the Remote Build REXX exec, CLZTLFTP, on a number of occasions to perform the following steps:
   a. Check to see if the directories specified in the CLZTULOW member exist,
   b. If they don't exist, run a MKDIR on the FTP Server to create them.
   c. Copy the Java source to the FTP Server directories.
   d. The translator then builds the Java compile shell from the CLZTJAVW control member, using the CLZTCPTW control member to define the class locations. This is used to tell Java where, within the SCLM hierarchy, included class files can be found. The output from this process is put into a file called makefile.bat, which is sent to the FTP Server.
   e. A SITE EXEC command is issued for the makefile.bat to start the Java compile.

f. The translator waits until a file that signals that the compile has finished has been created. It then goes off to the FTP server to get the listing and class files that have been created. These are copied into SCLM.

Translators and translator control files

This section lists all of the translator and translator control files that you modify during the installation process.

Remote Build and Deploy S-JDK JCL members

CLZTRUID
JCL to create encrypted user ID and password for Remote FTP Server

Remote Build and Deploy S-JDK translators

CLZTJBIZ
Translator for e-Business objects

CLZTJAVA
Translator for JAVA

CLZTJAR
Translator for JAR

Remote Build and Deploy S-JDK translator control files

CLZTJARW
Input to FTP CLZTJAR Translator; compile shell for Windows JAR type

CLZTJAVW
Input to FTP CLZTJAVA Translator; compile shell for Windows JAVA type

CLZTCPTW
Input to FTP JAVA and JAR translators; %classpath% substitution

CLZTULOW
Input to all S-JDK FTP translators; SCLM-to-Windows directory mapping rules

CLZTHTPD
Input to FTP CLZTJAVA translator; ADDTYPE list for Java compiles

Remote Build and Deploy S-JDK translator execs

CLZTLFTP
Input to FTP JAVA and JAR Translators; FTP processing REXX exec

A step-by-step approach

This section provides an overview of the steps involved in installing the S-JDK for Remote Build and Deploy.

Table 25. S-JDK for FTP/RBD installation steps

Before you begin	
1.	Review system and software requirements.
2.	Determine inventory values and type definitions.

Table 25. S-JDK for FTP/RBD installation steps (continued)

CP1.	Verify steps as shown in "Checkpoint #1 for S-JDK for FTP/RBD installation" on page 116.
Customize translators and translator control files.	
3.	Review and modify all translators and translator control file members.
CP2.	Verify steps as shown in "Checkpoint #2 for S-JDK FTP/RBD installation" on page 137
Define the S-JDK inventory, USS and Cloud 9 parts.	
4.	Modify and run CLZTALIB, CLZTAVSM, and CLZTPDEF to build S-JDK Project Definitions.
5.	Modify and run CLZC9J06 to define S-JDK types to Cloud 9
6.	Modify and run CLZTAUNX to define USS directories
7.	Review CLZJIBM UNIX Shell
CP3.	Verify steps as shown in "Checkpoint #3 for S-JDK for FTP/RBD installation" on page 149.

Chapter 12. Before you begin

This section describes the preparation steps that you undertake before starting the installation of the S-JDK for Remote Build and Deploy feature.

Step 1: Review software and assumptions

In this step, you review the system and software requirements for S-JDK for FTP/RBD installation.

System requirements

To successfully install Cloud 9 S-JDK for RBD, the following system requirements must be in place at your installation:

Table 26. System Requirements

z/OS Operating System	Version 2 Release 7 (or higher)
SCLM	Standard z/OS
IBM Cloud 9	Cloud 9 installed and configured
FTP Server	Installed and configured on a computer remote to the mainframe where Cloud 9 is installed
Java compiler	Installed on the computer running the FTP Server

FTP server prerequisites

In order for the Cloud 9 solution to communicate and perform builds on a remote server, that server must be able to support the following FTP commands:

- ASCII
- BINARY
- CD
- DELETE
- GET
- LCD
- LIST
- MKDIR
- PUT
- QUIT
- QUOTE SITE EXEC

To ensure that these commands are available, you can run the following JCL to display the FTP commands that can be invoked on your remote server:

```
//FTPCHECK JOB
(#ACCT),'FTP-CHECK',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//STEP1   EXEC PGM=FTP
//SYSIN   DD   *
9.123.456.78
ftpuser
password
```

```
HELP
HELP SITE
QUIT
//SYSPRINT  DD   SYSOUT=*
```

QUOTE SITE EXEC is used by the Java Remote Build translator, CLZTJAVA, and by the Jar Remote Build translator, CLZTJAR. The deployment translators do not use the QUOTE SITE command.

You might need to check your specific FTP server documentation to determine if the SITE EXEC command is supported. A number of FTP servers do not support the SITE EXEC command.

You need to tailor the sample REXX translator supplied with this solution (CLZTLFTP) to send commands and recognize replies from your FTP server. Our testing and configuration was done using a popular industry FTP server, Serv-U from RhinoSoft.

Java prerequisites

If you are planning on running Java compiles or Jar processes on the FTP server, you need to ensure that javac.exe and jar.exe are installed on that server. These are available in the Java 2 SDK for the platform on which you intend to build. You must be able to invoke the javac.exe program through a batch file (Windows) or command file (UNIX or Linux).

Assumptions

The installation administrator understands how to define types to SCLM.

The installation administrator understands how to configure the required FTP server on the target FTP platforms. The task of configuring the FTP servers might already be completed at this point. If not, contact your network system administrator to arrange for the the work to be performed.

The installation administrator understands REXX. The CLZTLFTP REXX Script might need to be modified.

Step 2: Determine inventory values and type definitions

Determine SCLM and remote server inventory values

The Remote Build and Deploy S-JDK is set up with default values. These default values are meant to be modified throughout the various translators, HTML and JCL members. Before making modifications to these members, review the following worksheets and fill in your site specific inventory values.

It is important to view the SCLM Inventory and the Remote Server directory structure as extensions to each other. Review both tables before making decisions about the values.

SCLM inventory value worksheet

Table 27. SCLM Inventory Value Worksheet

SCLM Inventory Name	Default Value	Your Values
Project	IBMDEMO	
Alt-project	IBMDEMO	
Group	DEV,QA,REL	
Types	JAVA,JAR,HTML,GRAPHICS	
Languages	JAVA,JAR,EBIZ	

Remote server directory value worksheet

Table 28. Remote Server Directory Value Worksheet

Remote Mapping Locations	Default Value	Your Values
Listings	• /ibmdemo/dev/listings • /ibmdemo/qa/listings • /ibmdemo/rel/listings	
Classes	• /ibmdemo/dev/classes • /ibmdemo/qa/classes • /ibmdemo/rel/classes	
Graphics	• /ibmdemo/dev/graphics • /ibmdemo/qa/graphics • /ibmdemo/rel/graphics	
Html	• /ibmdemo/dev/html • /ibmdemo/qa/html • /ibmdemo/rel/html	
Jar	• /ibmdemo/dev/jar • /ibmdemo/qa/jar • /ibmdemo/rel/jar	

Review SCLM and Cloud 9 type definitions

This step documents which types need to be defined to SCLM and Cloud 9. Before moving on to the translator and control file modification, review all types selected for Remote Build and Deploy S-JDK:

1. First, fill in each unique type in Table 18.
2. Then, for each type, review the definition in SCLM and Cloud 9 SLR for existence, consistency, and correctness.
3. If the types aren't defined yet, there is another step for updating the project definition and you can include the type definition process there (Chapter 14, "Define the S-JDK inventory, remote FTP server and Cloud 9 parts," on page 139).

Note: The following matrix is filled in with the default values.

Type review matrix

Table 29. Type Review Matrix

Type	Language	Defined to SCLM	Defined to Cloud 9 SLR	Case Sensitive Yes/No	Binary Yes/No	Lrecl	Recfm
Java 1	Java			Yes	No	256	VB
Jar	Jar			Yes	Yes	256	VB
Html 1	Ebiz			Yes	No	256	VB
Graphics	Ebiz			Yes	Yes	256	VB
Javaclas	N/A			Yes	Yes	256	VB
Javalist	N/A			Yes	No	256	VB
See "Determining LRECL and file attributes" on page 115							

Determining that types exist

To determine if a type exists, go to Cloud 9 and use the List SCLM Files command to access the SCLM Query page. Select a Group then examine the list of Types and verify that the Types are there. If they do not exist, then you must define the Types to SCLM and allocate the data sets in ISPF.

Determining Cloud 9 definitions

To determine if the SCLM type is defined to the Cloud 9 SLR (long name registry), run the IVP JCL member CLZC9J06. Review the output from the job, verifying that the SCLM type has been defined with the proper attributes. If not, modify this job to define the types to the Cloud 9 SLR.

Determining LRECL and file attributes

To determine the Logical Record Length (LRECL) of the type, go into SCLM Option 3.2 and display the data set information for one of the Type data sets. For e-business types, the LRECL is 256 and the RECFM = VB.

Note: If there are records in your e-Business Source types, such as HTML, that extend beyond 256 bytes, you need to assign a library with a larger LRECL in order to stop truncation. For the purpose of this installation guide, an assumption has been made that your record lengths do not exceed 256 bytes.

Checkpoint #1 for S-JDK for FTP/RBD installation

At this point the following tasks should be completed.

Table 30. Checkpoint #1 for S-JDK for RBD installation

Data Set Names	Completed?
Review all SCLM Inventory Default Values	
Review all Remote Server Directory Default Values	
Determine actual SCLM Inventory and Remote Server Directory Values	
Determine Remote Build and Deploy S-JDK Type Matrix	
Document SCLM Types and Cloud 9 definitions.	

Chapter 13. Customize translators and translator control files

This part describes the processes to be undertaken in customizing the components used to perform remote builds and deployments to a remote server. This involves the modification of JCL members, SCLM translator files, control files and the FTP Rexx exec.

Step 3: Review and modify JCL, translators, control files and execs

Step 3a. Review and modify remote FTP server user ID generation job

In this step, you review and modify the CLZTRUID member found in CLZ.SCLZJCL. This job returns an encrypted user ID and password that is used as input to the CLZTULOW control file, to allow building and deployment to a remote server. Before running this step, you must have configured your FTP server to have a user ID and password that Cloud 9 can use to access the Remote FTP Server. On your FTP server, you must also have defined the home directory and drive to which Cloud 9 has access.

```
//  JOBCARD
//* ------------------------------------------------
//*
//* MEMBER: CLZTRUID
//* EXAMPLE OF USERID/PASSWORD GENERATION FOR CLOUD 9 FTP SERVER
//* DEPLOYMENT AND JAVA/JAR COMPILER SUPPORT
//*
//* ------------------------------------------------
//STEP1   EXEC PGM=CLZTFILE
//STEPLIB   DD DSN=CLZ.SCLZLOAD,DISP=SHR
//SYSIN    DD   *,DLM='/?'
/* REXX */

TRUE=1; FALSE=0
DEBUG=FALSE

/* DEVELOPMENT MACHINE */
UIDPW = 'USER1,672MR6'
RESULT = CLZTLJVS(DEBUG,'ENCODE',UIDPW)
SAY 'LOGIN=WWW.DEVBOX.COM,' || RESULT

/* QA MACHINE */
UIDPW = 'QAUSER,BIRTHDAY'
RESULT = CLZTLJVS(DEBUG,'ENCODE',UIDPW)
SAY 'LOGIN=192.168.254.77,' || RESULT
/* PRODUCTION MACHINE */
UIDPW = 'BERNIE,SUCHAGOODDOG'
RESULT = CLZTLJVS(DEBUG,'ENCODE',UIDPW)
SAY 'LOGIN=WWW.PRODMACHINE.COM,' || RESULT
RETURN 0
/?
//SYSOUT   DD   DSN=&&REXX(PASSWORD),DISP=(NEW,PASS),
//    UNIT=SYSDA,SPACE=(TRK,(10,10,10)),
//    DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
//* ------------------------------------------------
//STEP2   EXEC PGM=IKJEFT01,DYNAMNBR=400
//STEPLIB   DD DSN=CLZ.SCLZLOAD,DISP=SHR
//SYSTSIN   DD *
%PASSWORD
//SYSTSPRT  DD SYSOUT=*
//SYSEXEC   DD DSN=&&REXX,DISP=(OLD,DELETE)
```

Figure 40. CLZTRUID — Remote FTP Server userid generation job

To create the encrypted user ID/password fields:

1. Using ISPF EDIT, access member CLZTRUID in the SCLZJCL data set.
2. Copy your job card values to the top of the member.
3. Modify the STEPLIB to reflect your installation of Cloud 9.
4. Modify the UIDPW = to have the user ID and password of the computer with which you are going to comunicate.
5. Modify the LOGIN= to be the IP address or IP name of the computer with which you are communicating.
6. Submit the job.

Note: This job should end with COND CODE=0. If it does not:

a. Review your job card parameters and the JCL for errors.
b. Resubmit the job.

The job output you receive should look something like this:

```
READY
%PASSWORD
LOGIN=9.190.173.70 111424CFC386FFE053DFD387DFE502CFC183CFD723DD64
READY
END
```

Figure 41. Job output from CLZTRUID

7. You can then use your mouse to cut the LOGIN line and paste it into the CLZTULOW member when you are tailoring it.

Step 3b. Review and modify translators

Review and modify Java build translator CLZTJAVA

In this step, you review and modify the CLZTJAVA member found in the CLZ.SCLZJCL library. Copy this member to your project definition source and update it to reference the control files listed in "Step 3c. Review and modify translator control files" on page 128. Ensure that the FTP control files are the ones that are specified in the translator at this time.

The Cloud 9 Java Translator utilizes the following functions: PARSE (save), BUILD, and COPY (promote). The Language is specified as JAVA in the translator, however, if you are going to utilise both the S-JDK for USS build and the S-JDK for RBD build, this can be changed to a different value.

The **PARSE function** causes the Java source to be scanned before being saved in SCLM.

The **BUILD function** consists of two steps. In Step 1, CLZTRJV1 copies all files contained in the SCLM package that need to be copied to the target server location before invocation of step 2 (the compile step). The UNIXLOC file consists of FTP login information and SCLM-to-Remote FTP Server location mapping rules. The HTTPD file contains EBCDIC-to-ASCII conversion rules.

The Java compile (JAVAC) is invoked in Step 2, CLZTRJVC, of the BUILD function. The ddname FILEIN contains the source to be compiled. The ddname JCOMPILE contains a shell script that is tailored before invoking the Java compiler. The CLASSPTH file is read and CLASSPATH statements are inserted in the JAVAC shell script. The UNIXLOC SCLM-to-Remote FTP Server mapping file is used to send the source, classes and compiled listing output to the correct location on the target server. When the compile completes, generated class files are written to file CLASSES. The JAVAC compiler output is written to JAVALIST.

Notes:

1. Only the first user ID and matching location in the UNIXLOC file is used when performing the Java compile.
2. By changing the FLMALLOC for DDNAME=SYSPRINT in the second BUILD step of the JAVA language translator to have a PRINT=Y, the JAVAC output is written out to ddname BLDLIST when a non-zero return code is set. The current statement has PRINT=N.
3. The Cloud 9 Java translator will use the first server in the UNIXLOC file that matches the login information. If the server is not available, the compile will fail, and no JAVA classes or listing will be stored in SCLM.

The **COPY function**, CLZTRJVC, deploys Java source to target locations when a SCLM Promote action is invoked. The COPY function can be removed if you do

not want to deploy Java source during a Promote action. FILEIN contains the Java source to be deployed. UNIXLOC defines the target location where the source is copied. The HTTPD file contains EBCDIC-to-ASCII conversion rules.

Build Map usage

In the sample translator, CLZTJAVA, Java classes are written to the ddname CLASSES and the listing file is written to the ddname JAVALIST.

For Java class files, more than one Java class file can be created as an output of the compile process. The related SCLM member names are different to the input source member name. Also, the generated Java listing has a different name than the Java source.

The SCLM translator CLZTJAVA uses the IOTYPE=P for the ddnames CLASSES and JAVALIST.

The Build Map for the Java source, as created during the SCLM Build function, contains a list of all Java classes and the name of the Java listing file. These output components of the Java compile process are checked by SCLM only during a SCLM Promote function, but not on a SCLM Build function. Therefore, specifying MODE=CONDITIONAL on a SCLM Build does not cause the Java source to be recompiled when the associated Java class files or listing file are deleted.

Validate the integrity of the Build Map by using the Promote function and specifying MODE=REPORT. If the report shows that your Build Map does not match the Build Map outputs that you have defined, you need to perform a Build function against the Java source using the MODE=FORCE option.

Control files are described later in this document.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*-------------------------------------------------------------------*
*  NAME:     CLZTJAVA                                               *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE = JAVA, LANGUAGE = JAVA.     *
*-------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR REQUIRES THAT FLMALLOC STATEMENTS BE      *
*         CUSTOMIZED DEPENDING ON WHETHER YOU ARE RUNNING USS OR    *
*         FTP. ALSO VARIOUS INPUT FILES MAY ALSO NEED TO BE         *
*         CUSTOMIZED.  BOTH CONTROL INPUT AND REXX EXECUTABLES ARE  *
*         READ FROM THE PRODUCT LIBRARY CLZ.SCLZCGI.                *
*-------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING   *
*         CONTROL FILES:                                            *
*          CLZ.SCLZCGI(CLZTCPTH)                        USS         *
*          CLZ.SCLZCGI(CLZTCPTW)                        FTP         *
*          CLZ.SCLZCGI(CLZTULOC)                        USS         *
*          CLZ.SCLZCGI(CLZTULOW)                        FTP         *
*          CLZ.SCLZCGI(CLZTHTPD)                                    *
*          CLZ.SCLZCGI(CLZTJAVC)                 USS                *
*          CLZ.SCLZCGI(CLZTJAVW)                 FTP SITE COMMAND   *
*-------------------------------------------------------------------*
*        JAVACLAS AND JAVALIST MUST BE DEFINED AS TYPES TO THE      *
*        PROJDEFS LOAD MODULE.                                      *
*                                                                   *
* CHANGE LOG: Z021120C/A0696 - CHANGE INVOCATION OF REXX            *
*             Z231104A/M0268 - ALTER DDNAME TO DSNAME REFERENCES    *
*-------------------------------------------------------------------*
         FLMLANGL    LANG=JAVA,VERSION=TEXTV1.0
         FLMTRNSL FUNCTN=PARSE,                                     C
               COMPILE=FLMLPGEN,                                    C
               PORDER=1,                                            C
               OPTIONS=(LANG=T,                                     C
               LISTINFO=@@FLMLIS,                                   C
               LISTSIZE=@@FLMSIZ,                                   C
               SOURCEDD=SOURCE,                                     C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
* ------------------------------------------------------------------ *
*                     BUILD TRANSLATOR                              *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                     C
               CALLNAM='COPY PKG MEMBERS',                          C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJV1,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               PDSDATA=Y
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                    USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                    FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
```

Figure 42. CLZTJAVA — S-JDK Translator for JAVA (Part 1 of 2)

```
* ------------------------------------------------------------------ *
*                     BUILD TRANSLATOR                               *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                        C
               CALLNAM='INVOKE JAVAC',                                 C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJVC,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               OPTIONS='@@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMBR JAC
               VACLAS JAVALIST'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=JCOMPILE,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTJAVC)            USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTJAVW)            FTP SITE COMMAND
         FLMALLOC DDNAME=CLASSPTH,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTCPTH)                      USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTCPTW)                      FTP
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                      USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                      FTP
         FLMALLOC DDNAME=JAVACLAS,DFLTTYP=JAVACLAS,LANG=EBIZ,          C
               LRECL=256,BLKSIZE=27998,RECFM=VB,RECNUM=60000,          C
               IOTYPE=P,KEYREF=OUT1,DIRBLKS=20
         FLMALLOC DDNAME=JAVALIST,DFLTTYP=JAVALIST,LANG=EBIZ,          C
               LRECL=256,BLKSIZE=27998,RECFM=VB,RECNUM=60000,          C
               IOTYPE=P,KEYREF=OUT2
         FLMALLOC DDNAME=SYSPRINT,IOTYPE=O,RECFM=FBA,LRECL=121,        C
               RECNUM=2500,PRINT=N
* ------------------------------------------------------------------ *
*                    PROMOTE TRANSLATOR                              *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=COPY,                                         C
               CALLNAM='JAVA PROMOTE',                                 C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJVP,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               PDSDATA=Y,                                              C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG'
         FLMALLOC DDNAME=SYSEXEC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                    USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                    FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ------------------------------------------------------------------ *
```

Figure 42. CLZTJAVA — S-JDK Translator for JAVA (Part 2 of 2)

Review and modify Jar build translator CLZTJAR

In this step, you review and modify the CLZTJAR member, found in the CLZ.SCLZJCL library delivered with the SMP/E installation of Cloud 9. Copy this member to your project definition source and update it to reference the control files listed in "Step 3c. Review and modify translator control files" on page 128. Also ensure that the FTP control files are the ones that are specified in the translator at this time.

The Cloud 9 Jar Translator utilizes the following functions: PARSE (save), BUILD, and COPY (promote). The Language is specified as JAR in the translator, however, if you are going to utilise both the Jar USS build and the Jar FTP build, this can be changed to a different value.

The **PARSE function** causes Cloud 9 Jar control statements to be scanned before being saved in SCLM.

The **BUILD function**, CLZTRJAR, invokes the Jar compiler. The ddname FILEIN specifies the Jar directives specifying the location of the files to be included in the Jar. The ddname JARCOMP contains a shell script that is tailored before invoking the Jar compiler. The UNIXLOC SCLM-to-Remote FTP Server rule mapping file is used to send the tailored shell to a specific target location. This file defines the location of the input files and the listing output for this Jar compile. When the compile completes, generated jar file is written to ddname JAR. The Jar compiler output is written to JARLIST.

Note: Only the first user ID and matching location in the UNIXLOC file is used when performing the Jar compile.

The **COPY function**, CLZTRJVC, deploys Jar source to target locations when a SCLM Promote action is invoked. FILEIN contains the Jar file to be deployed. UNIXLOC defines the target location where the source is copied. The HTTPD file contains EBCDIC-to-ASCII conversion rules.

Control files are described later in this document.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*-------------------------------------------------------------------*
*  NAME:     CLZTJAR                                                *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE = JAR, LANGUAGE = JAR.       *
*-------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR REQUIRES THAT FLMALLOC STATEMENTS BE      *
*         CUSTOMIZED DEPENDING ON WHETHER YOU ARE RUNNING USS OR    *
*         FTP. ALSO VARIOUS INPUT FILES MAY ALSO NEED TO BE         *
*         CUSTOMIZED.  BOTH CONTROL INPUT AND REXX EXECUTABLES ARE  *
*         READ FROM PRODUCT LIBRARY CLZ.SCLZCGI.                    *
*-------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING   *
*         CONTROL FILES:                                            *
*          CLZ.SCLZCGI(CLZTULOC)                          USS       *
*          CLZ.SCLZCGI(CLZTULOW)                          FTP       *
*          CLZ.SCLZCGI(CLZTHTPD)                                    *
*          CLZ.SCLZCGI(CLZTJARU)                          USS       *
*          CLZ.SCLZCGI(CLZTJARW)                          FTP       *
*-------------------------------------------------------------------*
*        JAR AND JARLIST MUST BE DEFINED AS TYPES TO THE            *
*        PROJDEFS LOAD MODULE.                                      *
*                                                                   *
* CORRECTION: Z021120C/A0696 - CHANGE INVOCATION OF REXX            *
* D RICHARD : Z230825A/A0960 - CHANGE ALLOCATION OF JARLIST         *
* D RICHARD : Z230915A/Z230825A - REWORK WITH CORRECT VERSION       *
*-------------------------------------------------------------------*
         FLMLANGL    LANG=JAR,VERSION=TEXTV1.0
         FLMTRNSL FUNCTN=PARSE,                                      C
               COMPILE=FLMLPGEN,                                     C
               PORDER=1,                                             C
               OPTIONS=(LANG=T,                                      C
               LISTINFO=@@FLMLIS,                                    C
               LISTSIZE=@@FLMSIZ,                                    C
               SOURCEDD=SOURCE,                                      C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
```

Figure 43. CLZTJAR — S-JDK Translator for JAR (Part 1 of 2)

```
* ------------------------------------------------------------------ *
*                     BUILD TRANSLATOR                               *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=BUILD,                                        C
               CALLNAM='BUILD JAR',                                    C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJAR,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               OPTIONS='@@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMBR JAC
               R JARLIST'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=JARCOMP,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTJARU)                        USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTJARW)                        FTP
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                        USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                        FTP
         FLMALLOC DDNAME=JAR,IOTYPE=P,PRINT=Y,RECNUM=60000,            C
               RECFM=VB,BLKSIZE=27998,                                 C
               DFLTTYP=JAR,KEYREF=OBJ,LRECL=256,LANG=EBIZ
         FLMALLOC DDNAME=JARLIST,DFLTTYP=JARLIST,LANG=EBIZ,            C
               LRECL=256,BLKSIZE=27998,RECFM=VB,RECNUM=60000,          C
               IOTYPE=P,KEYREF=LIST
         FLMALLOC DDNAME=SYSPRINT,IOTYPE=O,RECFM=FBA,LRECL=121,        C
               RECNUM=2500,PRINT=N
* ------------------------------------------------------------------ *
*                    PROMOTE TRANSLATOR                              *
* ------------------------------------------------------------------ *
         FLMTRNSL FUNCTN=COPY,                                         C
               CALLNAM='PROMOTE JAR',                                  C
               CALLMETH=TSOLNK,                                        C
               COMPILE=CLZTRJVP,                                       C
               DSNAME=CLZ.SCLZCGI,                                     C
               PDSDATA=Y,                                              C
               OPTIONS='BINARY @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLC
               MMBR @@FLMTOG'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                        USS
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                        FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ------------------------------------------------------------------ *
```

Figure 43. CLZTJAR — S-JDK Translator for JAR (Part 2 of 2)

Using the JAR translator

The Cloud 9 Jar control statements are stored in SCLM in a JARMAKE member. This member should include a file extension of either .CMD or .BAT (This is dependent upon the target operating system, WINDOWS=.BAT, USS=.CMD). There are two supported variations of this standard:

- JARMAKEFILE.JAR.CMD (or .BAT) - In this variation, the .CMD/.BAT portion will be removed to create the resulting JAR name (JARMAKEFILE.JAR)
- JARMAKEFILE.CMD (or .BAT) - In this variation, the .CMD/.BAT portion will be overlaid with .JAR to create the resulting JAR name (JARMAKEFILE.JAR)

The invocation member (CLZTRJAR) in the SCLZCGI library may now pass an additional option (KEEPCMD=YES|NO) to CLZTLJAR. If KEEPCMD=NO (the default action), the temporary command file used to create the jar is deleted after it has been used. If KEEPCMD=NO, the temporary command file is retained on the remote system. The option should only be used for debugging purposes, as the command file is not in the build map.

Sample Jar make file

The JARMAKE member can use either absolute pathing or relative pathing (relative from location of the JAR file, which depends upon the directory that the remote build translator deploys the JARMAKE member. For example, given this directory tree:

```
/DEV
    |
    /CLASSES
    /GRAPHICS
    /JAVA
```

and that the resulting JAR file is to be stored in /DEV/JAVA, the contents of the JARMAKE member could be:

```
/DEV/CLASSES/Classname.CLASS          (Absolute path)
../CLASSES/ Class2.CLASS              (Relative path)
../GRAPHICS/Picture.GIF               (Relative path)
```

Another example of the jar process is to create a jar file using a combination of locations and specific files. Our sample file will build a jar file composed of the contents of several relative directory structures. The directory structures are relative to the directory where the jarmake file is placed. This location is controlled by the contents of the file associated with the unixloc ddname. For our example, we have associated the following file with an SCLM type of JARMAKE and a language of JARMAKE. The file has a name of TESTME.BAT. The name of the jar file created by this process will be the member name with the extension of .jar overlaid on it. For this example, that means the jar name will be TESTME.JAR.

The TESTME.BAT contents:

```
org\apache\bcel
org\IBM\install
graphics\mouse.jpg
control\stuff.htm
```

The JARMAKE file has four lines that all begin in column 1. The jar will be composed of the directory structure `org\apache\bcel` and all of its sub-directories; `org\IBM\install` and its sub-directories; and the two individual files, `graphics\mouse.jpg`, and `control\stuff.htm`.

Review and modify e-Business translator CLZTJBIZ

In this step, you review and modify the CLZTJBIZ member found in the CLZ.SCLZJCL library delivered with the SMP/E installation of Cloud 9. Copy this member to your project definition source and update it to reference the control files listed in “Step 3c. Review and modify translator control files” on page 128. Also ensure that the FTP control files are the ones that are specified in the translator at this time.

The Cloud 9 e-Business translator utilizes the following functions: PARSE (save), BUILD and COPY (promote). The Language of EBIZ is specified in the translator.

The **PARSE function** causes SCLM to parse input before saving e-business objects in SCLM.

The **BUILD and COPY function**, CLZTRJVP, invokes the Cloud 9 e-business deployment program. The ddname FILEIN contains the file to be deployed. The UNIXLOC file consists of the SCLM-to-Remote FTP Server location mapping rules. The HTTPD file contains EBCDIC-to-ASCII conversion rules. File extensions associated with files added with the language EBIZ must be defined to the UNIXLOC file. If the file extension is not found in the UNIXLOC file, parameter 2 of this translator specifies the default to be used by this translator.

Control files are described later in this document.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
*--------------------------------------------------------------------*
*  NAME:     CLZTJBIZ    (CLZ.SCLZJCL)                               *
*  PURPOSE:  S-JDK TRANSLATOR FOR TYPE = GRAPHICS, HTML, XML        *
*            LANGUAGE = EBIZ.                                        *
*--------------------------------------------------------------------*
*  NOTES: THIS TRANSLATOR DOES NOT REQUIRE CUSTOMIZATION, HOWEVER,  *
*         THE VARIOUS INPUT FILES DO. BOTH CONTROL INPUT AND REXX   *
*         EXECUTABLE CODE ARE READ IN FROM THE PRODUCT LIBRARY      *
*         CLZ.SCLZCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECS.    *
*--------------------------------------------------------------------*
*         YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING   *
*         CONTROL FILES:                                            *
*          CLZ.SCLZCGI(CLZTULOC)                          USS       *
*          CLZ.SCLZCGI(CLZTULOW)                          FTP       *
*          CLZ.SCLZCGI(CLZTHTPD)                                    *
*--------------------------------------------------------------------*
         FLMLANGL    LANG=EBIZ,VERSION=TEXTV1.0
         FLMTRNSL FUNCTN=PARSE,                                     C
               COMPILE=FLMLPGEN,                                    C
               PORDER=1,                                            C
               OPTIONS=(LANG=T,                                     C
               LISTINFO=@@FLMLIS,                                   C
               LISTSIZE=@@FLMSIZ,                                   C
               SOURCEDD=SOURCE,                                     C
               STATINFO=@@FLMSTP)
         FLMALLOC DDNAME=SOURCE,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
*
```

Figure 44. CLZTJBIZ — S-JDK Translator for EBIZ (Part 1 of 2)

```
------------------------------------------------------------------ *
*                  BUILD TRANSLATOR                                *
* ---------------------------------------------------------------- *
         FLMTRNSL FUNCTN=BUILD,                                     C
               CALLNAM='UNIX BUILD',                                C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJVP,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG @@FLMBIO'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                    USS
           FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                    FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ---------------------------------------------------------------- *
*                  PROMOTE  TRANSLATOR                             *
* ---------------------------------------------------------------- *
         FLMTRNSL FUNCTN=COPY,                                      C
               CALLNAM='UNIX PROMOTE',                              C
               CALLMETH=TSOLNK,                                     C
               COMPILE=CLZTRJVP,                                    C
               DSNAME=CLZ.SCLZCGI,                                  C
               PDSDATA=Y,                                           C
               OPTIONS='TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@FLMMC
               BR @@FLMTOG'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZCGI
         FLMALLOC DDNAME=FILEIN,IOTYPE=A
           FLMCPYLB @@FLMDSN(@@FLMMBR)
         FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
* **       FLMCPYLB CLZ.SCLZCGI(CLZTULOC)                    USS
           FLMCPYLB CLZ.SCLZCGI(CLZTULOW)                    FTP
         FLMALLOC DDNAME=HTTPD,IOTYPE=A
           FLMCPYLB CLZ.SCLZCGI(CLZTHTPD)
* ---------------------------------------------------------------- *
```

Figure 44. CLZTJBIZ — S-JDK Translator for EBIZ (Part 2 of 2)

Step 3c. Review and modify translator control files

CASE SENSITIVITY ALERT!

The following translator control files contain case sensitive data. To ensure that the values are not automatically changed to uppercase during editing, issue the 'CAPS OFF' command on the command line of your ISPF session.

Modify CLZTULOW — common SCLM-to-remote server life cycle mapping rules

In this step, you review and modify the CLZTULOW member found in the SCLZCGI library. This member is input to all Remote Build and Deploy S-JDK translators. It is used to map the SCLM-to-Remote FTP server life cycles locations. This control file is used in the Build, Promote and Delete Process.

```
* ------------------------------------------------------------- *
* CLOUD 9 JAVA S-JDK COMPONENT      (NT Sample)                 *
* ------------------------------------------------------------- *
* NAME:    CLZTULOW                                           *
* PURPOSE: SCLM TO directory life cycle mapping rules.        *
* REFER:   DIRECTLY REFERENCED IN TRANSLATOR DDNAME UNIXLOC.  *
* ------------------------------------------------------------- *
* prj,alt,grp,typ            FTP location       KEEP or DELETE source
*                                               on promote
* Development machine
LOGIN=www.dev1machine.com,1102268F82439F89299F86879F568F349F68,SLASH=\,
     HOME=C:\HOMEDIR 1
LOGIN=192.168.254.22,1102268F82439F89299F86879F568F349F68              *
IBMDEMO,IBMDEMO,DEV,GRAPHICS /ibmdemo/dev/graphics       DELETE PERM=777
IBMDEMO,IBMDEMO,DEV,HTML     /ibmdemo/dev/html           DELETE PERM=777
IBMDEMO,IBMDEMO,DEV,JAVA     /ibmdemo/dev/java           DELETE PERM=777
IBMDEMO,IBMDEMO,DEV,JAVACLAS /ibmdemo/dev/classes        DELETE PERM=777
IBMDEMO,IBMDEMO,DEV,JAVALIST /ibmdemo/dev/cout           DELETE PERM=777
IBMDEMO,IBMDEMO,DEV,JARMAKE  /ibmdemo/dev                DELETE PERM=777
IBMDEMO,IBMDEMO,DEV,JAR      /ibmdemo/dev/jar            DELETE PERM=777
IBMDEMO,IBMDEMO,DEV,JARLIST  /ibmdemo/dev/cout           DELETE PERM=777
*
IBMDEMO,IBMDEMO,QA,GRAPHICS  /ibmdemo/qa/graphics        DELETE PERM=777
IBMDEMO,IBMDEMO,QA,HTML      /ibmdemo/qa/html            DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAVA      /ibmdemo/qa/java            DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAVACLAS  /ibmdemo/qa/classes         DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAVALIST  /ibmdemo/qa/cout            DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JARMAKE   /ibmdemo/qa                 DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAR       /ibmdemo/qa/jar             DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JARLIST   /ibmdemo/qa/cout            DELETE PERM=777
*
IBMDEMO,IBMDEMO,REL,GRAPHICS /ibmdemo/rel/graphics DELETE PERM=777
IBMDEMO,IBMDEMO,REL,HTML     /ibmdemo/rel/html           DELETE PERM=777
IBMDEMO,IBMDEMO,REL,JAVA     /ibmdemo/rel/java           DELETE PERM=777
IBMDEMO,IBMDEMO,REL,JAVACLAS /ibmdemo/rel/classes        DELETE PERM=777
IBMDEMO,IBMDEMO,REL,JAVALIST /ibmdemo/rel/cout           DELETE PERM=777
IBMDEMO,IBMDEMO,REL,JARMAKE  /ibmdemo/rel                DELETE PERM=777
IBMDEMO,IBMDEMO,REL,JAR      /ibmdemo/rel/jar            DELETE PERM=777
IBMDEMO,IBMDEMO,REL,JARLIST  /ibmdemo/rel/cout           DELETE PERM=777
*
* ------------------------------------------------------------------
* QA machine
LOGIN=192.168.254.11,0108269F83798F89228F91549F45F7F89F68
*
IBMDEMO,IBMDEMO,QA,GRAPHICS  /qa/ibmdemo/qa/graphics     DELETE PERM=777
IBMDEMO,IBMDEMO,QA,HTML      /qa/ibmdemo/qa/html         DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAVA      /qa/ibmdemo/qa/java         DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAVACLAS  /qa/ibmdemo/qa/classes      DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAVALIST  /qa/ibmdemo/qa/cout         DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JARMAKE   /qa/ibmdemo/qa              DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JAR       /qa/ibmdemo/qa/jar          DELETE PERM=777
IBMDEMO,IBMDEMO,QA,JARLIST   /qa/ibmdemo/qa/cout         DELETE PERM=777
```

1 This line is a continuation of the previous line. It has been broken over two lines for display purposes only.

Figure 45. CLZTULOW — SCLM TO directory life cycle mapping rules

This control file provides several functions to Cloud 9 translators. First, it is used to define FTP login information. Second, it defines the SCLM-to-Remote mapping. Third, it is used to set file access permissions for files sent to servers. Fourth, it defines whether files are to be deleted from their source locations (source and target) when an SCLM Promote is requested.

The LOGIN statement is generated by running the utility CLZTRUID, covered previously. In the FTP example in the member provided, deployment for the DEV group sends data to two servers: www.dev1machine.com and 192.168.254.22. If promotion into QA occurs, files are sent to the FTP server 192.168.254.11.

The parameters specified on the LOGIN= statement are seperated by commas and are as follows:

- IP Address or IP Name of the FTP server
- Encoded user ID and password created by running the utility JCL CLZTRUID.
- **SLASH=** parameter (optional). Use this parameter to specify the direction of the slash for path names. Windows uses a backward slash (\) and UNIX and Linux operating systems use a forward slash (/).

  If omitted, a forward slash (UNIX) is assumed.
- **HOME=** parameter (optional). This parameter is required if you have mapped the home directory on your FTP server to a location other than C:\. Do not specify this parameter for FTP servers running UNIX or Linux unless you also tailor the files CLZTJAVW and CLZTJARW to include the HOME= specification.

  For the Java and Jar BUILD functions, only the first FTP server with a matching SCLM location is used. In the FTP example given in the member provided, compiles are only performed on www.dev1machine.com.

For the mapping statements the definitions are as follows:

Position 1
SCLM Life Cycle location

Position 2
Remote Server Life Cycle location

Position 3
Dispostion of files on promote

Position 4
Permissions to be allocated to files created in the specified directory. This parameter is used to give different UNIX permissions to files of different types and also to files at different levels in the hierarchy.

The CLZTULOW rules can be set up to allow deployment to multiple servers, as illustrated in Figure 46 on page 131. This figure shows that files being promoted into the SCLM Group of QA will be deployed to two separate machines. Assuming that the file being promoted is stored with an SCLM Type of GRAPHICS, the file would be deployed to the directory `/ibmdemo/qa/graphics` on the DEV1 machine and to `/qa/ibmdemo/qa/graphics` on the QA machine. Deployment rules can also be established to cause files to be copied or propagated to servers that correspond to lower levels within an SCLM hierarchy.

Consider the following hierarchical definition:

```
path 1: DEV1 -> TEST1 -> QA -> REL
path 2: DEV2 -> TEST2 -> QA -> REL
```

Now consider that a separate server has been assigned to each directory location. Files are to be propagated to the corresponding servers (and servers lower in the hierarchical path) as a SCLM promote action is performed. So a file promoted from DEV1 to TEST1 would be copied to the corresponding servers of DEV1 and TEST1. A file promoted from TEST1 to QA would be propagated to servers DEV1, TEST1,

QA, DEV2, and TEST2. The following rules illustrate how to propagate files to lower locations in the SCLM hierarchy.

```
LOGIN=dev1.cigi.net,1003027377392882188823FB8,SLASH=\,HOME=C:\
IBMDEMO,IBMDEMO,DEV1,GRAPHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,TEST1,GRPAHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,QA,GRAPHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,REL,GRAPHICS /ibmdemo/graphics KEEP PERM=777
LOGIN=dev2.cigi.net,1228BA32442162,SLASH=\,HOME=C:\
IBMDEMO,IBMDEMO,DEV2,GRAPHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,TEST2,GRPAHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,QA,GRAPHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,REL,GRAPHICS /ibmdemo/graphics KEEP PERM=777
LOGIN=test1.cigi.net,1003027377392882188823FB8,SLASH=\,HOME=C:\
IBMDEMO,IBMDEMO,TEST1,GRPAHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,QA,GRAPHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,REL,GRAPHICS /ibmdemo/graphics KEEP PERM=777
LOGIN=test2.cigi.net,4452BC5778CDEA7724865,SLASH=\,HOME=C:\
IBMDEMO,IBMDEMO,TEST2,GRPAHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,QA,GRAPHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,REL,GRAPHICS /ibmdemo/graphics KEEP PERM=777
LOGIN=qa.cigi.net,AAA562773091008277BC3488B,SLASH=\,HOME=C:\
IBMDEMO,IBMDEMO,QA,GRAPHICS /ibmdemo/graphics KEEP PERM=777
IBMDEMO,IBMDEMO,REL,GRAPHICS /ibmdemo/graphics KEEP PERM=777
LOGIN=rel.cigi.net,CCB1128837774000019888388,SLASH=\,HOME=C:\
IBMDEMO,IBMDEMO,REL,GRAPHICS /ibmdemo/graphics KEEP PERM=777
```

Figure 46. Example CLZTULOW propagation

Using the example CLZTULOW, a graphics file being promoted from QA to REL would be propagated to all servers in the hierarchy. In the event a file exists lower in the hierarchy, the propagation of files will stop at the point in the hierarchy where the first file was found to exist. For example, in a parallel development situation where a file resides in groups TEST1 and TEST2, and the file in TEST1 is promoted to QA, then the file being promoted will be propagated to DEV1, TEST1 and QA. It will not be propagated to DEV2 and TEST2 since a file exists lower in the path `DEV2 -> TEST2 -> QA -> REL`.

Modify CLZTCPTW — common %CLASSPATH% substitution

In this step, you review and modify the CLZTCPTW member found in the SCLZCGI library. This member is input to both the Remote Build and Deploy Java and Jar compile shells. It is used to fill in the %Classpath% variable in the compiler shells.

This is the SCLM hierarchy class path allocation control member. It tells the Java compile where to find additional class files that are within the SCLM hierarchy. The Java source libraries are included in the concatenation to accommodate the way that Java works. If Java come across a class file that it doesn't have in the class directory, it looks in the source directory for the Java source. The Java compiler then compiles the source to create the required class file. When the source directory is included as part of the class path allocation, Java can find the source in those cases where the class file is not in the class path.

```
* ------------------------------------------------------------- *
*  CLOUD 9 JAVA ftp support                                     *
* ------------------------------------------------------------- *
* NAME:    CLZTCPTW                                            *
* PURPOSE: #CLASSPATH# SUBSTITUTION FILE.                      *
* REFER:   DIRECTLY REFERENCED IN TRANSLATOR DDNAME CLASSPTH   *
* ------------------------------------------------------------- *
* prj,alt,gr            classpath concatenation
IBMDEMO,IBMDEMO,DEV     \ibmdemo\dev\classes
IBMDEMO,IBMDEMO,DEV     \ibmdemo\dev\java
IBMDEMO,IBMDEMO,DEV     \ibmdemo\qa\classes
IBMDEMO,IBMDEMO,DEV     \ibmdemo\qa\java
IBMDEMO,IBMDEMO,DEV     \ibmdemo\rel\classes
IBMDEMO,IBMDEMO,DEV     \ibmdemo\rel\java
```

Figure 47. CLZTCPTW — %CLASSPATH% Substitution File

Note: The actual class path might be more complex than one shown in Figure 47. For instance, the core Java class libraries might reside in directories outside of the standard SCLM/Remote FTP Server life cycle.

Modify CLZTJAVW — Java compile shell using FTP SITE EXEC

In this step, you review and modify the CLZTJAVW member found in the SCLZCGI library. This member is input to the CLZTJAVA translator for the Java Type. (You might need to review the path location with your Remote FTP Server Administrator.) This file is the template used to invoke the Java compile. The template is modified with the #CLASSPATH# statement which is substituted with data taken from CLZTCPTW. The parameters %1, %2 and %3 are substituted, based on rules defined in the CLZTULOW file as follows:

- %1 = Java source location
- %2 = Java file name
- %3 = Java class location
- %4 = tempory dataset name used to determine if the build process has finished on the remote server

The symbolic parms denoted by %1, %2, %3, and %4 can represent long directory names and long file names. To create the correct syntax when these long names contain spaces or apostrophes, the symbolics should be surrounded by quotation marks, for example:

```
"%1" "%2" "%3""%4"
```

This syntax can also be used when the directory and file names do not contain spaces or apostrophes.

```
rem ---------------------------------------------------------------
rem CLOUD 9 JAVA s-jdk component    (nt sample)
rem  ---------------------------------------------------------------
rem  NAME:    CLZTJAVW
rem  PURPOSE: JAVA COMPILE SHELL
rem  REFER:   DIRECTLY REFERENCED IN THE TRANSLATOR DDNAME JCOMPILE.
rem  ---------------------------------------------------------------
set JAVA_HOME="\jdk1.3.1_02\"
set PATH=%java_home%\bin
set CLASSPATH=#CLASSPATH#;%CLASSPATH%
cd %1
javac -verbose -d %3 %2
copy a b >%4
```

Figure 48. CLZTJAVW — JAVA compile shell using FTP SITE EXEC

Note: The actual class path might be more complex than one shown in Figure 48. For instance, the core Java class libraries might reside in directories outside of the standard SCLM/Remote FTP Server life cycle.

Modify CLZTJARW — Jar compile shell

In this step, you review and modify the CLZTJARW member found in the SCLZCGI library. This member is the default input to the CLZTJAR translator for the JAR Type. The template is modified with the #CLASSPATH# statement substituted with data taken from CLZTCPTW. The Jar commands are appended to the template before invocation of the Jar compiler.

```
# ------------------------------------------------------------- #
# CLOUD 9 JAR template ftp deployment (nt example)              #
# ------------------------------------------------------------- #
# NAME:    CLZTJARW                                             #
# PURPOSE: JAR COMPILE SHELL                                    #
# REFER:   DIRECTLY REFERENCED IN TRANSLATOR DDNAME JARCOMP.    #
* ------------------------------------------------------------- #
set JAVA_HOME="\jdk1.3.1_02\"
set PATH=%java_home%\bin
```

Figure 49. CLZTJARW — JAR Compile Shell

Modify CLZTHTPD — Addtype list for Java compile

In this step, you review and modify the CLZTHTPD member found in the SCLZCGI target library. This member contains Addtype definitions, similar to those that can be found in the CLZHTTPD member in the SCLZHTML target library. Addtype definitions are used by the Java compile process to determine if objects included in the Java compile are Binary or Text. This is required by the copy function that is part of the compile process.

```
#--------------------------------------------------------------------
# Name: CLZTHTPD
# Purpose: Cloud9 Server Rules File
# Usage: This file is used by the Java compile process
#--------------------------------------------------------------------
#--------------------------------------------------------------------
#
#Non-standard MIME types declared here. (User style MIME types)
#
#--------------------------------------------------------------------
AddType .asm text/asm ebcdic 1.0 # Assemble Macros
AddType .doc binary/doc binary 1.0 # Microsoft Word Documents
AddType .ppt binary/ppt binary 1.0 # Power Point Documents
AddType .cob text/cobol ebcdic 1.0 # COBOL Source Code
AddType .cbl text/cobol ebcdic 1.0 # COBOL Source Code
AddType .cobol text/cobol ebcdic 1.0 # COBOL Source Code
#--------------------------------------------------------------------
#
AddType .cer application/x-x509-user-cert ebcdic 0.5 # Browser Certificate
AddType .der application/x-x509-ca-cert binary 1.0 # CA Certificate
AddType .mime www/mime binary 1.0 # Internal -- MIME is
AddType .bin application/octet-stream binary 1.0 # Uninterpreted binary
AddType .class application/octet-stream binary 1.0 # Java applet or application
AddType .pdf application/pdf binary 1.0
AddType .ai application/postscript ebcdic 0.5 # Adobe Illustrator
AddType .PS application/postscript ebcdic 0.8 # PostScript
AddType .eps application/postscript ebcdic 0.8
AddType .ps application/postscript ebcdic 0.8
AddType .rtf application/x-rtf ebcdic 1.0 # RTF
AddType .csh application/x-csh ebcdic 0.5 # C-shell script
AddType .latex application/x-latex ebcdic 1.0 # LaTeX source
AddType .cdf application/x-cdf ebcdic 1.0 # Channel Definition Format
AddType .sh application/x-sh ebcdic 0.5 # Shell-script
AddType .tcl application/x-tcl ebcdic 0.5 # TCL-script
AddType .tex application/x-tex ebcdic 1.0 # TeX source
AddType .t application/x-troff ebcdic 0.5 # Troff
AddType .roff application/x-troff ebcdic 0.5
AddType .tr application/x-troff ebcdic 0.5
AddType .man application/x-troff-man ebcdic 0.5 # Troff with man macros
AddType .me application/x-troff-me ebcdic 0.5 # Troff with me macros
AddType .ms application/x-troff-ms ebcdic 0.5 # Troff with ms macros
AddType .gtar application/x-gtar binary 1.0 # Gnu tar
AddType .shar application/x-shar ebcdic 1.0 # Shell archive
AddType .wrl x-world/x-vrml binary 1.0 # VRML
AddType .snd audio/basic binary 1.0 # Audio
AddType .au audio/basic binary 1.0
AddType .aiff audio/x-aiff binary 1.0
AddType .aifc audio/x-aiff binary 1.0
AddType .aif audio/x-aiff binary 1.0
AddType .wav audio/x-wav binary 1.0 # Windows+ WAVE format
AddType .bmp image/bmp binary 1.0 # OS/2 bitmap format
AddType .gif image/gif binary 1.0 # GIF
AddType .ief image/ief binary 1.0 # Image Exchange fmt
AddType .jpg image/jpeg binary 1.0 # JPEG
AddType .JPG image/jpeg binary 1.0
AddType .JPE image/jpeg binary 1.0
AddType .jpe image/jpeg binary 1.0
AddType .JPEG image/jpeg binary 1.0
AddType .jpeg image/jpeg binary 1.0
AddType .tif image/tiff binary 1.0 # TIFF
AddType .tiff image/tiff binary 1.0
AddType .ras image/cmu-raster binary 1.0
```

Figure 50. CLZTHTPD — Cloud 9 Server Rules file (Part 1 of 2)

```
AddType .pnm image/x-portable-anymap binary 1.0 # PBM Anymap format
AddType .pbm image/x-portable-bitmap binary 1.0 # PBM Bitmap format
AddType .pgm image/x-portable-graymap binary 1.0 # PBM Graymap format
AddType .ppm image/x-portable-pixmap binary 1.0 # PBM Pixmap format
AddType .rgb image/x-rgb binary 1.0
AddType .xbm image/x-xbitmap ebcdic 1.0 # X bitmap
AddType .xpm image/x-xpixmap binary 1.0 # X pixmap format
AddType .xwd image/x-xwindowdump binary 1.0 # X window dump (xwd)
AddType .html text/html ebcdic 1.0 # HTML
AddType .htm text/html ebcdic 1.0 # HTML on PCs
AddType .htmls text/x-ssi-html ebcdic 1.0 # Server-side includes
AddType .shtml text/x-ssi-html ebcdic 1.0 # Server-side includes
AddType .c text/plain ebcdic 0.5 # C source
AddType .h text/plain ebcdic 0.5 # C headers
AddType .C text/plain ebcdic 0.5 # C++ source
AddType .cc text/plain ebcdic 0.5 # C++ source
AddType .hh text/plain ebcdic 0.5 # C++ headers
AddType .java text/plain ebcdic 0.5 # Java source
AddType .js text/plain ebcdic 0.5 # JavaScript source
AddType .m text/plain ebcdic 0.5 # Objective-C source
AddType .f90 text/plain ebcdic 0.5 # Fortran 90 source
AddType .txt text/plain ebcdic 0.5 # Plain text
AddType .bat text/plain ebcdic 0.5 # Plain text
AddType .css text/css 8bit 1.0 # W3C Cascading Style Sheets
AddType .rtx text/richtext ebcdic 1.0 # MIME Richtext format
AddType .tsv text/tab-separated-values ebcdic 1.0 # Tab-separated values
AddType .etx text/x-setext ebcdic 0.9 # Struct Enchanced Txt
AddType .MPG video/mpeg binary 1.0 # MPEG
AddType .mpg video/mpeg binary 1.0
AddType .MPE video/mpeg binary 1.0
AddType .mpe video/mpeg binary 1.0
AddType .MPEG video/mpeg binary 1.0
AddType .mpeg video/mpeg binary 1.0
AddType .qt video/quicktime binary 1.0 # QuickTime
AddType .mov video/quicktime binary 1.0
AddType .avi video/x-msvideo binary 1.0 # MS Video for Windows
AddType .movie video/x-sgi-movie binary 1.0 # SGI moviepalyer
AddType .zip multipart/x-zip binary 1.0 # PKZIP
AddType .tar multipart/x-tar binary 1.0 # 4.3BSD tar
AddType .ustar multipart/x-ustar binary 1.0 # POSIX tar
AddType *.* www/unknown binary 0.2 # Try to guess
AddType * www/unknown binary 0.2 # Try to guess
AddType .cxx text/plain ebcdic 0.5 # C++
AddType .for text/plain ebcdic 0.5 # Fortran
AddType .mar text/plain ebcdic 0.5 # MACRO
AddType .log text/plain ebcdic 0.5 # logfiles
AddType .com text/plain ebcdic 0.5 # scripts
AddType .sdml text/plain ebcdic 0.5 # SDML
AddType .list text/plain ebcdic 0.5 # listfiles
AddType .lst text/plain ebcdic 0.5 # listfiles
AddType .def text/plain ebcdic 0.5 # definition files
AddType .conf text/plain ebcdic 0.5 # definition files
AddType . text/plain ebcdic 0.5 # files with no extension
AddType .JP932 text/x-DBCS binary 1.0 IBM-932 # Japanese DBCS
AddType .JPeuc text/x-DBCS binary 1.0 IBMeucJP # Japanese DBCS
```

Figure 50. CLZTHTPD — Cloud 9 Server Rules file (Part 2 of 2)

Position 1
Addtype keyword

Position 2
file extension

Position 3
MIME type and subtype you want to bind to files that match the corresponding suffix pattern.

Position 4
The MIME content encoding to which the data has been converted.

Position 5
An optional indicator of relative value (on a scale of 0.0 to 1.0) for the content type.

Position 6
Description to associate with the document For more information about the Addtype directives, see the *HTTP Server Planning, Installing, and Using* manual (SC34-4826-00).

Step 3d. Review and modify translator execs

Modify CLZTLFTP — FTP communication program

Communication between Cloud 9 translators and FTP servers is performed by a Rexx routine, delivered as source code. The program, CLZTLFTP, performs basic FTP functions such as GET, PUT, MKDIR, Send Site command and DELETE. In addition, this program is called before allocation of semi-permanent files used when invoking the FTP process. You have the ability to override the default high-level qualifier (the user ID) for data sets used during the FTP process. These files are deleted when the Cloud 9 translator has completed execution.

The program CLZTLFTP expects the results returned from your FTP server to be in a specific syntax. If the FTP server that you are running returns results in a different syntax, you might need to modify CLZTLFTP. This program also needs to be modified if you are planning on replacing FTP with an alternative file transport protocol, such as MQSeries.

Java tracing

The tracing facility for the Java process aids with debugging problems. It can be helpful to see if the tailoring of the Java control members has been carried out properly. The tracing can be turned on in the following modules, found in the SCLZCGI target library: CLZTRJAR, CLZTRJVC, CLZTRJVP, CLZTRJV1 and CLZTRJDL. Each of these members contains two calls to the same program, one of which uses the DEBUG parameter and is commented out. To activate the tracing, change the commenting to use the call with the DEBUG parameter. The default is to have tracing turned off.

Checkpoint #2 for S-JDK FTP/RBD installation

At this point you should have completed the following tasks:

Table 31. Checkpoint #2 for S-JDK/RBD installation

Task	Completed?
Reviewed commands allowable on the FTP server with which you are communicating	
Reviewed and modified Userid generation JCL CLZTRUID	
Reviewed and modified JAVA translator CLZTJAVA	
Reviewed and modified JAR translator CLZTJAR	
Reviewed and modified e-Business translator CLZTJBIZ	
Reviewed and modified the SCLM to Remote FTP Server mapping control file CLZTULOW	
Reviewed and modified class path control file CLZTCPTW	
Reviewed and modified FTP JAVA compile shell CLZTJAVW	
Reviewed and modified JAR compile shell CLZTJARW	
Reviewed and modified FTP Communication program CLZTLFTP	

Chapter 14. Define the S-JDK inventory, remote FTP server and Cloud 9 parts

Step 4: Update the Project Definition

To complete the enabling of the S-JDK for RBD, the current SCLM Project Definition must be updated with both the S-JDK types and the S-JDK translators. These definitions must be included with the current project definitions that define your existing project or used as a standalone SCLM Project. The following section shows the Type definitions and Translator copy statements required to define the S-JDK defaults. Typically, these types and translators are included in a common project. The following member shows a stand-alone project definition JCL stream, containing the type and translator definitions. The JCL member, CLZTPDEF, can be found in the CLZ.SCLZJCL library. This is meant as an example. With your SCLM administrator, decide whether the new types are to be an isolated project or part of an existing project.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
//* (JOBCARD)
//*--------------------------------------------------------------------*
//*  NAME:     CLZTPDEF                                                *
//*  PURPOSE:  S-JDK STANDALONE PROJECT DEFINITION.                    *
//*--------------------------------------------------------------------*
//*  TO USE THIS JCL YOU MUST:                                         *
//*         1) INSERT A VALID JOB CARD.                                *
//*         2) REVIEW THE SCLM VALUES - THEY ARE INITIALLY SET TO      *
//*            THE S-JDK DEFAULT VALUES.                               * 0
//*         3) REVIEW THE DATASET NAMES - THEY ARE INITIALLY SET TO    *
//*            THE S-JDK DEFAULT VALUES.                               *
//*         4) MODIFY THE DATASET NAMES TO MATCH ACTUAL CHOSEN S-JDK   *
//*            TYPES.                                                  *
//*         4) MODIFY THE SCLM VALUES TO MATCH ACTUAL CHOSEN S-JDK     *
//*            TYPES.                                                  *
//*         5) REVIEW AND MODIFY THE S-JDK TRANSLATORS AND LANGUAGE    *
//*            NAMES TO THE CHOSE TYPES.                               *
//*         6) CHANGE THE UNIT=TDISK TO THE APPROPRIATE UNIT FOR       *
//*            TEMPORARY FILES.                                        *
//COMP   PROC
//*--------------------------------------------------------------------*
//STEP1ASM EXEC PGM=ASMA90,REGION=3072K,COND=(0,NE),
//            PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT)'
//SYSLIB   DD  DSN=CLZ.SCLZJCL,DISP=SHR
//         DD  DSN=ISP.SISPMACS,DISP=SHR
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
//SYSPRINT DD  SYSOUT=*
//    PEND
//*--------------------------------------------------------------------*
```

Figure 51. CLZTPDEF — S-JDK Standalone Project Definition (Part 1 of 3)

```
//LINK   PROC
//STEP2LNK EXEC PGM=IEWL,REGION=2048K,COND=(0,NE),
//             PARM='LIST,XREF,LET,RENT,REUS,CALL'
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DSN=IBMDEMO.PROJDEFS.OBJLIB,DISP=SHR
//SYSLMOD  DD  DSN=IBMDEMO.PROJDEFS.LOAD,DISP=SHR
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
//*-------------------------------------------------------------------*
//    PEND
//COMPILE   EXEC COMP
//SYSIN   DD *
         TITLE '*** PROJECT DEFINITION FOR PROJECT=IBMDEMO ***'
IBMDEMO  FLMABEG
*
*        ************************************************************
*        *  DEFINE THE AUTHORIZATON CODES                           *
*        ************************************************************
GRPDEV   FLMAGRP AC=(DEV)
*
*        ************************************************************
*        *  DEFINE THE TYPES                                        *
*        ************************************************************
COBOL    FLMTYPE ,                    HOST COBOL CODE
DOC      FLMTYPE ,                    WORD FOR WINDOWS DOCUMENTATION
GRAPHICS FLMTYPE ,                    GRAPHICS FILES (JPG, GIF)
HTML     FLMTYPE ,                    HTML AND JAVASCRIPT
JAR      FLMTYPE ,                    JAVA JAR
JAVA     FLMTYPE ,                    JAVA SOURCE
JAVACLAS FLMTYPE ,                    JAVA CLASSES
JAVALIST FLMTYPE ,                    JAVA LISTING
PACKAGES FLMTYPE ,                    PACKAGE ARCHHL
*
*        ************************************************************
*        *  DEFINE THE GROUPS                                       *
*        ************************************************************
DEV      FLMGROUP AC=(DEV),KEY=Y,PROMOTE=QA      DEVELOPMENT
QA       FLMGROUP AC=(DEV),KEY=Y,PROMOTE=REL     QUALITY ASSURANCE
REL      FLMGROUP AC=(DEV),KEY=Y                 PRODUCTION
*
*********************************************************************
*               PROJECT PROJDEFSS
*********************************************************************
        FLMCNTRL ACCT=IBMDEMO.PROJDEFS.ACCT,                          X
              VERS=IBMDEMO.PROJDEFS.VERSION,                          X
              XREF=IBMDEMO.PROJDEFS.XREF,                             X
              LIBID=SCLM
*
*********************************************************************
*               VERSIONING AND AUDITIBILITY                         *
*********************************************************************
        FLMATVER GROUP=DEV,TYPE=COBOL,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=DOC,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=GRAPHICS,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=HTML,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAR,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAVA,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAVACLAS,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=JAVALIST,VERSION=YES
        FLMATVER GROUP=DEV,TYPE=PACKAGES,VERSION=YES
```

Figure 51. CLZTPDEF — S-JDK Standalone Project Definition (Part 2 of 3)

```
*
        FLMATVER GROUP=QA,TYPE=COBOL,VERSION=YES
        FLMATVER GROUP=QA,TYPE=DOC,VERSION=YES
        FLMATVER GROUP=QA,TYPE=GRAPHICS,VERSION=YES
        FLMATVER GROUP=QA,TYPE=HTML,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAR,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAVA,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAVACLAS,VERSION=YES
        FLMATVER GROUP=QA,TYPE=JAVALIST,VERSION=YES
        FLMATVER GROUP=QA,TYPE=PACKAGES,VERSION=YES
*
        FLMATVER GROUP=REL,TYPE=COBOL,VERSION=YES
        FLMATVER GROUP=REL,TYPE=DOC,VERSION=YES
        FLMATVER GROUP=REL,TYPE=GRAPHICS,VERSION=YES
        FLMATVER GROUP=REL,TYPE=HTML,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAR,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAVA,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAVACLAS,VERSION=YES
        FLMATVER GROUP=REL,TYPE=JAVALIST,VERSION=YES
        FLMATVER GROUP=REL,TYPE=PACKAGES,VERSION=YES
*
************************************************************************
*               LANGUAGE DEFINITION TABLES                             *
************************************************************************
*             TRANSLATOR           LANGUAGE
*        COPY  FLM@ARCD           -- ARCHDEF                    --
*        COPY  FLM@COB2           -- COBOL                      --
         COPY  CLZTJAR            -- JAR                        --
         COPY  CLZTJAVA           -- JAVA                       --
         COPY  CLZTJBIZ           -- E-BUSINESS OBJECTS         --
*
************************************************************************
         FLMAEND
/*
//SYSLIN DD DSN=IBMDEMO.PROJDEFS.OBJLIB(IBMDEMO),DISP=SHR
//LINK   EXEC LINK
//SYSLIN DD *
 INCLUDE SYSLIB(IBMDEMO)
 NAME IBMDEMO(R)
/*
```

Figure 51. CLZTPDEF — S-JDK Standalone Project Definition (Part 3 of 3)

Note: If you want to make use of both the S-JDK USS build and the S-JDK FTP build, you can create seperate translators for each build and use different language types in each. For example: instead of using LANG=JAVA in the CLZTJAVA translator, set up two translators, one of which specifies LANG=JAVAUSS and the other specifies LANG=JAVAFTP. The Language Definition Table in the Project definition must then include both of these translators.

Optionally, if you are building a new isolated project for the S-JDK system, you need to run two additional jobs: CLZTALIB and CLZTAVSM. The JCL stream, CLZTALIB, allocates all of the group libraries and CLZTAVSM allocates all of the SCLM VSAM files.

Step 4a: Allocate new S-JDK type data set

Allocate SCLM type files — CLZTALIB

Regardless of whether you build the types into an existing project or as a stand alone project, each type needs a set of libraries. The following JCL stream allocates these required libraries.

```
//* (JOBCARD)
//*-------------------------------------------------------------------*
//* NAME: CLZTALIB *
//* PURPOSE: DELETE AND ALLOCATE THE S-JDK BASE AND VERSION FILES. *
//*-------------------------------------------------------------------*
//* TO USE THIS JCL YOU MUST: *
//* 1) INSERT A VALID JOB CARD. *
//* 2) REVIEW THE DATASET NAMES - THEY ARE DELIVERED *
//* MATCHING THE S-JDK DEFAULT VALUES. *
//* 3) MODIFY THE DATASET NAMES TO MATCH ACTUAL CHOSEN S-JDK *
//* TYPES. *
//* 4) CHANGE THE UNIT=DUNIT TO THE APPROPRIATE UNIT FOR *
//* PERMANENT FILES. *
//*-------------------------------------------------------------------*
//* DELETE ALL S-JDK TYPE FILES *
//*-------------------------------------------------------------------*
//DELETE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE IBMDEMO.DEV.COBOL
DELETE IBMDEMO.DEV.DOC
DELETE IBMDEMO.DEV.GRAPHICS
DELETE IBMDEMO.DEV.HTML
DELETE IBMDEMO.DEV.JAR
DELETE IBMDEMO.DEV.JAVA
DELETE IBMDEMO.DEV.JAVACLAS
DELETE IBMDEMO.DEV.JAVALIST
DELETE IBMDEMO.DEV.JCL
DELETE IBMDEMO.DEV.PACKAGES
DELETE IBMDEMO.QA.COBOL
DELETE IBMDEMO.QA.DOC
DELETE IBMDEMO.QA.GRAPHICS
DELETE IBMDEMO.QA.HTML
DELETE IBMDEMO.QA.JAR
DELETE IBMDEMO.QA.JAVA
DELETE IBMDEMO.QA.JAVACLAS
DELETE IBMDEMO.QA.JAVALIST
DELETE IBMDEMO.QA.JCL
DELETE IBMDEMO.QA.PACKAGES
DELETE IBMDEMO.REL.COBOL
DELETE IBMDEMO.REL.DOC
DELETE IBMDEMO.REL.GRAPHICS
DELETE IBMDEMO.REL.HTML
DELETE IBMDEMO.REL.JAR
DELETE IBMDEMO.REL.JAVA
DELETE IBMDEMO.REL.JAVACLAS
DELETE IBMDEMO.REL.JAVALIST
DELETE IBMDEMO.REL.JCL
DELETE IBMDEMO.REL.PACKAGES
DELETE IBMDEMO.DEV.COBOL.VERSION
DELETE IBMDEMO.DEV.DOC.VERSION
DELETE IBMDEMO.DEV.GRAPHICS.VERSION
DELETE IBMDEMO.DEV.HTML.VERSION
DELETE IBMDEMO.DEV.JAR.VERSION
DELETE IBMDEMO.DEV.JAVA.VERSION
```

Figure 52. CLZTALIB — Delete and allocate S-JDK Base and Version Files (Part 1 of 3)

```
DELETE IBMDEMO.DEV.JAVACLAS.VERSION
DELETE IBMDEMO.DEV.JAVALIST.VERSION
DELETE IBMDEMO.DEV.JCL.VERSION
DELETE IBMDEMO.DEV.PACKAGES.VERSION
DELETE IBMDEMO.QA.COBOL.VERSION
DELETE IBMDEMO.QA.DOC.VERSION
DELETE IBMDEMO.QA.GRAPHICS.VERSION
DELETE IBMDEMO.QA.HTML.VERSION
DELETE IBMDEMO.QA.JAR.VERSION
DELETE IBMDEMO.QA.JAVA.VERSION
DELETE IBMDEMO.QA.JAVACLAS.VERSION
DELETE IBMDEMO.QA.JAVALIST.VERSION
DELETE IBMDEMO.QA.JCL.VERSION
DELETE IBMDEMO.QA.PACKAGES.VERSION
DELETE IBMDEMO.REL.COBOL.VERSION
DELETE IBMDEMO.REL.DOC.VERSION
DELETE IBMDEMO.REL.GRAPHICS.VERSION
DELETE IBMDEMO.REL.HTML.VERSION
DELETE IBMDEMO.REL.JAR.VERSION
DELETE IBMDEMO.REL.JAVA.VERSION
DELETE IBMDEMO.REL.JAVACLAS.VERSION
DELETE IBMDEMO.REL.JAVALIST.VERSION
DELETE IBMDEMO.REL.JCL.VERSION
DELETE IBMDEMO.REL.PACKAGES.VERSION
//*-------------------------------------------------------------------*
//* PROC TO ALLOCATE BASE FILES *
//*-------------------------------------------------------------------*
//ALLOC PROC FILE=
//STEP1 EXEC PGM=IEFBR14
//DD01 DD DSN=&FILE,
// DISP=(NEW,CATLG,DELETE),
// UNIT=DUNIT,
// SPACE=(CYL,(2,10,100)),
// DCB=(RECFM=VB,LRECL=256,BLKSIZE=0)
// PEND
//*-------------------------------------------------------------------*
//* PROC TO ALLOCATE VERSION FILES *
//*-------------------------------------------------------------------*
//VALLOC PROC FILE=
//STEP2 EXEC PGM=IEFBR14
//DD01 DD DSN=&FILE,
// DISP=(NEW,CATLG,DELETE),
// UNIT=DUNIT,
// SPACE=(CYL,(2,10,100)),
// DCB=(RECFM=VB,LRECL=350,BLKSIZE=0)
// PEND
//*-------------------------------------------------------------------*
//* NOW DO THE ALLOCATES *
//*-------------------------------------------------------------------*
//FILE01 EXEC ALLOC,FILE=IBMDEMO.DEV.COBOL
//FILE02 EXEC ALLOC,FILE=IBMDEMO.DEV.DOC
//FILE03 EXEC ALLOC,FILE=IBMDEMO.DEV.GRAPHICS
//FILE04 EXEC ALLOC,FILE=IBMDEMO.DEV.HTML
//FILE05 EXEC ALLOC,FILE=IBMDEMO.DEV.JAR
//FILE06 EXEC ALLOC,FILE=IBMDEMO.DEV.JAVA
//FILE07 EXEC ALLOC,FILE=IBMDEMO.DEV.JAVACLAS
```

Figure 52. CLZTALIB — Delete and allocate S-JDK Base and Version Files (Part 2 of 3)

```
//FILE08 EXEC ALLOC,FILE=IBMDEMO.DEV.JAVALIST
//FILE09 EXEC ALLOC,FILE=IBMDEMO.DEV.JCL
//FILE10 EXEC ALLOC,FILE=IBMDEMO.DEV.PACKAGES
//FILE11 EXEC ALLOC,FILE=IBMDEMO.QA.COBOL
//FILE12 EXEC ALLOC,FILE=IBMDEMO.QA.DOC
//FILE13 EXEC ALLOC,FILE=IBMDEMO.QA.GRAPHICS
//FILE14 EXEC ALLOC,FILE=IBMDEMO.QA.HTML
//FILE15 EXEC ALLOC,FILE=IBMDEMO.QA.JAR
//FILE16 EXEC ALLOC,FILE=IBMDEMO.QA.JAVA
//FILE17 EXEC ALLOC,FILE=IBMDEMO.QA.JAVACLAS
//FILE18 EXEC ALLOC,FILE=IBMDEMO.QA.JAVALIST
//FILE19 EXEC ALLOC,FILE=IBMDEMO.QA.JCL
//FILE20 EXEC ALLOC,FILE=IBMDEMO.QA.PACKAGES
//FILE21 EXEC ALLOC,FILE=IBMDEMO.REL.COBOL
//FILE22 EXEC ALLOC,FILE=IBMDEMO.REL.DOC
//FILE23 EXEC ALLOC,FILE=IBMDEMO.REL.GRAPHICS
//FILE24 EXEC ALLOC,FILE=IBMDEMO.REL.HTML
//FILE25 EXEC ALLOC,FILE=IBMDEMO.REL.JAR
//FILE26 EXEC ALLOC,FILE=IBMDEMO.REL.JAVA
//FILE27 EXEC ALLOC,FILE=IBMDEMO.REL.JAVACLAS
//FILE28 EXEC ALLOC,FILE=IBMDEMO.REL.JAVALIST
//FILE29 EXEC ALLOC,FILE=IBMDEMO.REL.JCL
//FILE30 EXEC ALLOC,FILE=IBMDEMO.REL.PACKAGES
//*
//FILE31 EXEC VALLOC,FILE=IBMDEMO.DEV.COBOL.VERSION
//FILE32 EXEC VALLOC,FILE=IBMDEMO.DEV.DOC.VERSION
//FILE33 EXEC VALLOC,FILE=IBMDEMO.DEV.GRAPHICS.VERSION
//FILE34 EXEC VALLOC,FILE=IBMDEMO.DEV.HTML.VERSION
//FILE35 EXEC VALLOC,FILE=IBMDEMO.DEV.JAR.VERSION
//FILE36 EXEC VALLOC,FILE=IBMDEMO.DEV.JAVA.VERSION
//FILE37 EXEC VALLOC,FILE=IBMDEMO.DEV.JAVACLAS.VERSION
//FILE38 EXEC VALLOC,FILE=IBMDEMO.DEV.JAVALIST.VERSION
//FILE39 EXEC VALLOC,FILE=IBMDEMO.DEV.JCL.VERSION
//FILE40 EXEC VALLOC,FILE=IBMDEMO.DEV.PACKAGES.VERSION
//FILE41 EXEC VALLOC,FILE=IBMDEMO.QA.COBOL.VERSION
//FILE42 EXEC VALLOC,FILE=IBMDEMO.QA.DOC.VERSION
//FILE43 EXEC VALLOC,FILE=IBMDEMO.QA.GRAPHICS.VERSION
//FILE44 EXEC VALLOC,FILE=IBMDEMO.QA.HTML.VERSION
//FILE45 EXEC VALLOC,FILE=IBMDEMO.QA.JAR.VERSION
//FILE46 EXEC VALLOC,FILE=IBMDEMO.QA.JAVA.VERSION
//FILE47 EXEC VALLOC,FILE=IBMDEMO.QA.JAVACLAS.VERSION
//FILE48 EXEC VALLOC,FILE=IBMDEMO.QA.JAVALIST.VERSION
//FILE49 EXEC VALLOC,FILE=IBMDEMO.QA.JCL.VERSION
//FILE50 EXEC VALLOC,FILE=IBMDEMO.QA.PACKAGES.VERSION
//FILE51 EXEC VALLOC,FILE=IBMDEMO.REL.COBOL.VERSION
//FILE52 EXEC VALLOC,FILE=IBMDEMO.REL.DOC.VERSION
//FILE53 EXEC VALLOC,FILE=IBMDEMO.REL.GRAPHICS.VERSION
//FILE54 EXEC VALLOC,FILE=IBMDEMO.REL.HTML.VERSION
//FILE55 EXEC VALLOC,FILE=IBMDEMO.REL.JAR.VERSION
//FILE56 EXEC VALLOC,FILE=IBMDEMO.REL.JAVA.VERSION
//FILE57 EXEC VALLOC,FILE=IBMDEMO.REL.JAVACLAS.VERSION
//FILE58 EXEC VALLOC,FILE=IBMDEMO.REL.JAVALIST.VERSION
//FILE59 EXEC VALLOC,FILE=IBMDEMO.REL.JCL.VERSION
//FILE60 EXEC VALLOC,FILE=IBMDEMO.REL.PACKAGES.VERSION
```

Figure 52. CLZTALIB — Delete and allocate S-JDK Base and Version Files (Part 3 of 3)

Step 4b (optional): Allocate new project VSAM files

Allocate project VSAM files — CLZTAVSM

If you are building an isolated, stand alone Project for the S-JDK types, there are three VSAM files that need to be allocated. The following JCL stream, found in the CLZ.SCLZJCL library, allocates these required libraries.

Those lines that might need to be modified are identified by being highlighted in Bold.

```
//* (JOBCARD)
//*-------------------------------------------------------------------*
//* NAME: CLZTAVSM *
//* PURPOSE: DELETE AND ALLOCATE THE S-JDK PROJECT VSAM FILES. *
//*-------------------------------------------------------------------*
//* TO USE THIS JCL YOU MUST: *
//* 1) INSERT A VALID JOB CARD. *
//* 2) REVIEW THE DATASET NAMES - THEY ARE DELIVERED *
//* MATCHING THE S-JDK DEFAULT VALUES. *
//* 3) MODIFY THE DATASET NAMES TO MATCH ACTUAL CHOSEN S-JDK *
//* TYPES. *
//* 4) CHANGE THE UNIT=TDISK TO THE APPROPRIATE UNIT FOR *
//* TEMPORARY FILES. *
//* 5) CHANGE THE "DVOLSER" VARIABLE TO THE APPROPRIATE *
//* VOLUME FOR VSAM FILES. *
//*********************************************************************
//* IBMDEMO.PROJDEFS.JCLLIB(ALLOVSAM)
//*********************************************************************
//ACCT1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE IBMDEMO.PROJDEFS.ACCT
DEFINE CLUSTER +
(NAME('IBMDEMO.PROJDEFS.ACCT') +
CYLINDERS(1 1) +
VOLUMES(DVOLSER) +
KEYS(26 0) +
RECORDSIZE(264 32000) +
SHAREOPTIONS(4,3) +
SPEED +
SPANNED +
UNIQUE) +
INDEX(NAME('IBMDEMO.PROJDEFS.ACCT.I') -
) +
DATA(NAME('IBMDEMO.PROJDEFS.ACCT.D') -
CISZ(2048) +
FREESPACE(50 50) +
)
/*
//*********************************************************************
//*
//* INITIALIZE THE ACCOUNTING FILE
//*
//*********************************************************************
//ACCT2 EXEC PGM=IDCAMS
//INPUT DD *
SCLM ACCOUNTING FILE INITIALIZATION RECORD
/*
//OUTPUT DD DSN=IBMDEMO.PROJDEFS.ACCT,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
88 Cloud 9 for SCLM for z/OS Installation Guide
```

Figure 53. CLZTAVSM — Delete and allocate S-JDK VSAM Files (Part 1 of 3)

```
//*********************************************************************
//* A JOB STEP IS THEN EXECUTED TO INITIALIZE THE FILE.
//*********************************************************************
//VERS1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE IBMDEMO.PROJDEFS.VERSION
DEFINE CLUSTER +
(NAME('IBMDEMO.PROJDEFS.VERSION') +
CYLINDERS(1 1) +
VOLUMES(DVOLSER) +
KEYS(40 0) +
RECORDSIZE(264 32000) +
SHAREOPTIONS(4,3) +
SPEED +
SPANNED +
UNIQUE) +
INDEX(NAME('IBMDEMO.PROJDEFS.VERSION.I') -
) +
DATA(NAME('IBMDEMO.PROJDEFS.VERSION.D') -
CISZ(2048) +
FREESPACE(50 50) +
)
/*
/*
//*********************************************************************
//* INITIALIZE THE AUDIT CONTROL FILE
//*********************************************************************
//VERS2 EXEC PGM=IDCAMS
//INPUT DD *
SCLM AUDIT CONTROL FILE INITIALIZATION RECORD
/*
//OUTPUT DD DSN=IBMDEMO.PROJDEFS.VERSION,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*********************************************************************
//* A JOB STEP IS THEN EXECUTED TO INITIALIZE THE FILE.
//*
//*********************************************************************
//XREF0 EXEC PGM=BZZFPARM,
// PARM='NEW,SCLM CROSS REFERENCE INITIALIZATION RECORD'
//STEPLIB DD DSN=IBMDEMO.PRODUCT.LOADLIB,DISP=SHR
//CIGPOUT DD DSN=&&PARMFILE,DISP=(NEW,PASS),
// UNIT=TDISK,DCB=(LRECL=128,BLKSIZE=12800,RECFM=FB),
// SPACE=(TRK,1)
//CIGLOG DD SYSOUT=*
//* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
//XREF1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
```

Figure 53. CLZTAVSM — Delete and allocate S-JDK VSAM Files (Part 2 of 3)

```
DELETE IBMDEMO.PROJDEFS.XREF
DEFINE CLUSTER +
(NAME('IBMDEMO.PROJDEFS.XREF') +
CYLINDERS(2 1) +
VOLUMES(DVOLSER) +
KEYS(128 0) +
RECORDSIZE(264 32000) +
SHAREOPTIONS(4,3) +
SPEED +
SPANNED +
UNIQUE) +
INDEX(NAME('IBMDEMO.PROJDEFS.XREF.I') -
) +
DATA(NAME('IBMDEMO.PROJDEFS.XREF.D') -
CISZ(2048) +
FREESPACE(50 50) +
)
/*
//*********************************************************************
//* INITIALIZE THE CROSS-REFERENCE FILE
//*********************************************************************
//XREF2 EXEC PGM=IDCAMS
//INPUT DD DSN=&&PARMFILE,DISP=(OLD,DELETE)
//OUTPUT DD DSN=IBMDEMO.PROJDEFS.XREF,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*
```

Figure 53. CLZTAVSM — Delete and allocate S-JDK VSAM Files (Part 3 of 3)

Step 5: Define S-JDK types to Cloud 9 SLR

The purpose of this step is to define the S-JDK types to your Cloud 9 SLR file. The following example shows the default S-JDK definitions required to define the S-JDK types to the Cloud 9 SLR. For information about the syntax of the SLR utility statements, see Appendix B, “Suite Long Name Registry,” on page 213.

Modify and submit CLZC9J06

1. Using ISPF EDIT, access member CLZC9J06 (CLZ.SCLZJCL library). This JCL should already have been modified during the base installation and IVP process.
2. Update the member, including the following SLR definitions (in BOLD)
3. Submit the job.

Note: This job should end with COND CODE=0.

```
//* (JOBCARD)
//* ------------------------------------------------------------------*
//* CLOUD 9 JAVA/S-JDK COMPONENTS. *
//* ------------------------------------------------------------------*
//* ------------------------------------------------------------------*
//* NAME: CLZC9J06 *
//* PURPOSE: DEFINE S-JDK TYPES TO THE SLR DATABASE. *
//* ------------------------------------------------------------------*
//* TO USE THIS JCL, YOU MUST: *
//* 1) INSERT A VALID JOB CARD WITH VALID CLASS AND REGION=0M *
//* 2) MAKE SURE THAT THE STEPLIB POINTS TO YOUR CLOUD9 *
//* AUTHORIZED DATASET THAT CONTAINS THE CIGINI. *
//* 3) MODIFY THE TYPE NAMES IF YOUHAVE CHANGED THE DEFAULT *
//* SCLM TYPES. *
//* ------------------------------------------------------------------ *
//STEP1 EXEC PGM=CZLSLR
//STEPLIB DD DSN=CLZ.SCLZLOAD,DISP=SHR
//CIGIN DD *
ADD NAME RULE FOR SCLM TYPE DOC CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE GRAPHICS CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE HTML CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE JAVA CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE JAR CASE SENSITIVE.
ADD NAME RULE FOR SCLM TYPE JAVACLAS CASE SENSITIVE.
//CIGLOG DD SYSOUT=*
```

Figure 54. CLZC9J06 — Define S-JDK Types to SLR Database

Java tracing: The tracing facility for the Java process aids with debugging problems and can be helpful to see if the tailoring of the Java control members has been carried out properly. The tracing can be turned on in the following modules, found in the SCLZCGI target library: CLZTRJVC, CLZTRJVP, CLZTRJV1 and CLZTRJDL. Follow the instructions in each member to determine the correct tracing operand. The default is to have tracing turned off.

Checkpoint #3 for S-JDK for FTP/RBD installation

At this point all SCLM and Cloud 9 definitions should be complete.

Table 32. Checkpoint #3 for S-JDK/RBD installation

Task	Completed?
Update your project definition with JDK Types and Translators (Optionally CLZTPDEF)	
Allocate New SCLM Type Libraries CLZTALIB	
Optionally allocate new SCLM project VSAM, only if creating a new project using CLZTAVSM	
Run CLZC9J06 to define S-JDK types to Cloud 9	

Define the S-JDK inventory, remote FTP server and Cloud 9 parts for S-JDK for FTP/RBD

Part 4. SCLM-FTP feature for remote deploy

Chapter 15. S-FTP installation overview

This part of the manual contains the installation procedure for the IBM Cloud 9 for SCLM for z/OS SCLM-File Transfer Protocol feature. This feature is a set of Remote Build and Deployment Scripts based in FTP. Within this manual, the following names are used to refer to these elements:

- IBM Cloud 9 for SCLM for z/OS is called ***Cloud 9***
- IBM Cloud 9 for SCLM for z/OS SCLM-File Transfer Protocol feature is called ***S-FTP***

The steps in this part are organized into four major sections:

- Before you begin
- Customize FTP translator and REXX script
- Create SCLM and Cloud 9 Definitions
- Perform Installation Verification Procedures

Overview of the S-FTP remote build and deploy

The S-FTP remote build and deploy provides you with a tailorable means with which to deploy code to remote FTP servers. By coding your own batch execution files, you can also perform remote builds on remote platforms. This feature has been largely superceeded by the S-JDK SCLM remote build and deploy Java Development Kit.

Modifying case-sensitive S-FTP values

During this installation, you modify several JCL members and REXX Scripts. Some of these files contain case-sensitive values. It is imperative that *before* modifying the files, you issue the CAPS OFF command to ensure that automatic upper casing does not occur. You are reminded of this case sensitivity issue where appropriate throughout this manual.

Product components used during installation

The following JCL members are modified during the installation process. They are located in SCLZJCL and are resident on the host. The names are provided here as an overview of naming standards and component functionality.

JCL members modified during installation:

CLZ@FTP1 Cloud 9 FTP translator

CLZRFTP1 REXX script

A step-by-step approach

Table 33. S-FTP installation steps

Before you begin	
1.	Review system, software and hardware considerations.
2.	**Review default inventory locations and USS locations**
2(a).	Review default S-FTP values and determine actual inventory and FTP targets to use.
2(b).	Review SLR and SCLM definitions for FTP supported types
CP1.	Verify steps as shown in “Checkpoint #1 for S-FTP installation” on page 159
Customize FTP translator and REXX script	
3.	Review and modify the FTP translator
4.	Review and modify the REXX script
Create SCLM and Cloud 9 definitions	
5.	Include CLZ@FTP1 in your SCLM Project Definition
CP2.	Verify steps as shown in “Checkpoint #2 for S-FTP installation” on page 171
Perform Installation Verification Procedures	
6.	Add an HTML file and perform a build on the file.

Chapter 16. Before you begin

Step 1: Review software and hardware requirements

In this step, you review the system, software and hardware requirements for S-FTP installation.

System requirements

To successfully install Cloud 9 S-FTP, the following system requirements must be in place at your installation:

Table 34. System requirements

z/OS Operating System	Version 2 Release 7 (or higher)
z/OS FTP Server	Standard z/OS
Target FTP Server	Specific to platform
IBM Cloud 9	Cloud 9 installed and configured

Assumptions

The installation administrator understands how to define types to SCLM.

The installation administrator understands how to configure FTP on the z/OS and target FTP platforms. The task of configuring the FTP servers might already be completed at this point. If this has not occurred, contact the network system administrator to perform the work.

The installation administrator has identified which Types are to be deployed to remote locations, and has defined them in the SCLM Project Definition and the SLR (for cross platform Types).

The installation administrator understands REXX. The CLZRFTP1 REXX Script needs to be modified. Although this manual provides guidelines, the modifications are best performed by someone who has REXX experience.

Step 2: Review default values and targets

Step 2(a): Review and set S-FTP inventory values

The S-FTP is delivered with default values. These default values are meant to be modified throughout the translator and REXX script. Before making modifications to these members, review the following worksheets and fill in your site specific Inventory Values. It is important to review and record all possible targets for builds and remote deployment.

SCLM inventory and REXX value worksheet

Table 35. SCLM Inventory and REXX Value Worksheet

SCLM inventory item	Variable name	Default value	Your values
SCLM group, used in CLZRFTP1	group	DEV, TEST	
SCLM 'make' type, referenced in CLZRFTP1	type	MAKE	
SCLM binary types, referenced as examples of types that have binary FTP attributes in CLZRFTP1	type	GIF, JPG	
SCLM language, set in CLZ@FTP1	lang	FTP1	
User, used in CLZRFTP1 in several places.	user	userid	
Directory, used in all four of the Set Target examples in CLZRFTP1.	dir	• For USS: '/u/userdir' • For ISP: '/isp/userdir' • For C drive: 'c:\userdir' • For A drive: 'a:\userdir'	
IP address, used in all four of the Set Target examples in CLZRFTP1.	ipaddr	999.999.999.99	
Port, used in all four of the Set Target examples in CLZRFTP1.	port	21	
Data set name for the FTP log file used at the end of CLZRFTP1.	userlog	userid.ftplog	

FTP target platform worksheet

Table 36. FTP target platform worksheet

Platform type	Type deployed	Group location deployed	IP address/port	User ID/ password	Target directory	Target directories defined? Yes/No	Target security? Yes/No	FTP server type at target
ISP - NT	HTML	TEST	999.999.999.999/21	Userid/ pass	/isp/userdir	Yes	No	Yes
Your Platform Here								

Step 2(b): Review SCLM and Cloud 9 type definitions

Before moving on to the REXX and Translator modification, review all types selected for deployment. First fill in each unique type in the Table 37. Then, for each type, review the definition in SCLM and the Cloud 9 SLR. Check for the type's existence, consistency and correctness.

Ensure that the FTP target directories are defined. Use Table 36 to document the target directories that are defined. You might also need to meet with your Network Administrator to review security for the target platforms.

Type review matrix

Table 37. Type Review Matrix

Type	Language is FTP1	Defined to SCLM	Defined to Cloud 9 SLR	Case Sensitive Yes/No	Binary (lrecl = 256)

Determine that types exist

To determine if a type exists:

1. Start Cloud 9 in your browser.
2. Click **LIST SCLM FILES**.
3. Select the SCLM project for which the FTP deployment is to be implemented.
4. Click the **?** button, for the Type field, then display the drop-down selection list.
5. Verify that the types in your worksheet are on this list.

If your types do not exist, you must define the types to SCLM and allocate the data sets in ISPF.

Determine that the language does not yet exist

In this process, you review and modify the FTP Translator member, CLZ@FTP1. This member defines the language as

```
LANG=FTP1
```

. If this definition is already in use for your Cloud 9 project, redefining it with this member might cause problems with the existing definition.

To determine if a language does not exist:

1. Start Cloud 9 in your browser.
2. Click **LIST SCLM FILES**.
3. Select the SCLM project for which the FTP deployment is to be implemented.
4. Click the **?** button, for the Language field, then display the drop-down selection list.
5. Verify that this list does not contain `FTP1`.

If `FTP1` is on the list, you can modify CLZ@FTP1 and change the language definition statement, for example, to

```
LANG=FTP2
```

.

Determine Cloud 9 definitions

To determine if the SCLM type is defined to the Cloud 9 SLR (long name registry):

1. Run the IVP JCL member CLZC9J06.
2. Review the output from the job, and verify that the SCLM has been defined with the proper attributes.

If not, modify this job to define the Types to Cloud 9.

Determining LRECL and file attributes

To determine the LRECL of the type, go into SCLM Option 3.2 and display the data set information for one of the type data sets. If the type is binary, the LRECL is 256 and the RECFM = VB.

Note: If there are records in your e-Business Source types, such as HTML, that extend beyond 256 bytes, you need to assign a library with a larger LRECL in order to stop truncation. For the purpose of this installation guide, an assumption has been made that your record lengths do not exceed 256 bytes.

Checkpoint #1 for S-FTP installation

At this point the following tasks should be completed.

Table 38. Checkpoint #1 for S-FTP installation

Data Set Names	Completed?
Review all SCLM and FTP Inventory Default Values	
Determine key SCLM target locations	
Determine Target Locations/Type Matrix	
Ensure that SCLM Types are properly defined to SCLM and Cloud 9.	
Ensure that the FTP target directories are defined.	

Chapter 17. Customize FTP translator and REXX script

Step 3: Review and modify CLZ@FTP1

In this step, you review and modify the CLZ@FTP1 member found in the **CLZ.SCLZJCL** file delivered with the SMP/E installation of Cloud 9. There is not a lot of modification for this member, the translator uses default IBM naming standards for all product files.

1. Using ISPF EDIT, access member CLZ@FTP1 in the CLZ.SCLZJCL library.
2. Ensure that the CLZ.SCLZJCL data set is in the SYSLIB concatenation of Project Definition ***JCL***. Either include the data set or copy this member to a library in the **SYSLIB** concatenation.

```
*********************************************************************
* NAME:    CLZ@FTP1                                                 *
* PURPOSE: INVOKE CLOUD 9 FTP REXX SCRIPT TO DEPLOY AND/OR BUILD    *
*          REMOTE OBJECTS. ( SCRIPT NAME IS CLZRFTP1)               *
*********************************************************************
* NOTE:    THIS PROTOTYPE TRANSLATOR REQUIRES CUSTOMIZATION         *
*          BY THE CUSTOMER OR INSTALLER.                            *
*          THE ACTUAL DEPLOYMENT OR BUILD REQUEST IS PERFORMED      *
*          BY THE REXX EXEC CLZRFTP1. THIS REXX EXEC NEEDS TO BE    *
*          CUSTOMIZED TO MEET THE INVENTORY NAMES AND TARGET        *
*          LOCATIONS REQUIRED BY THE CUSTOMER.                      *
*                                                                   *
*********************************************************************
* CHANGE ACTIVITY:                                                  *
* Z021220A JFP CHANGED LENGTH OF CIGPUNCH TO SUPPORT 250 SLR NAME   *
*                                                                   *
*********************************************************************
         FLMLANGL    LANG=FTP1,VERSION=TEXTV1.0
*********************************************************************
*
* PARSER TRANSLATOR
*
*********************************************************************
          FLMTRNSL   CALLNAM='SCLM TEXT PARSE',                     C
               FUNCTN=PARSE,                                        C
               COMPILE=FLMLPGEN,                                    C
               PORDER=1,                                            C
               OPTIONS=(SOURCEDD=SOURCE,                            C
               STATINFO=@@FLMSTP,                                   C
               LISTINFO=@@FLMLIS,                                   C
               LISTSIZE=@@FLMSIZ,                                   C
               LANG=T)
*          (* SOURCE      *)
           FLMALLOC  IOTYPE=A,DDNAME=SOURCE
            FLMCPYLB @@FLMDSN(@@FLMMBR)
```

Figure 55. CLZ@FTP1 S-FTP Translator (Part 1 of 3)

```
*********************************************************************
*
* BUILD TRANSLATOR
*
*  STEP 1A: GET THE LONGNAME FROM THE SLR
*
*********************************************************************
         FLMTRNSL FUNCTN=BUILD,CALLNAM='STEP1A: CLZFPARM',             X
               COMPILE=CLZFPARM,DSNAME=CLZ.SCLZLOAD,                   X
               OPTIONS='NEW,LIST LONGNAME WHERE SHORTNAME = @@FLMMBR.'
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W
         FLMALLOC DDNAME=CIGPOUT,IOTYPE=W
*                                        COPY CIGPOUT TO CIGIN
         FLMTRNSL FUNCTN=BUILD,CALLNAM='STEP1B: CLZFCOPY',             X
               COMPILE=CLZFCOPY,DSNAME=CLZ.SCLZLOAD,                   X
               OPTIONS='CIGPOUT ,CIGIN   '
         FLMALLOC DDNAME=CIGPOUT,IOTYPE=U
         FLMALLOC DDNAME=CIGIN,IOTYPE=W
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W,PRINT=Y
*                                        COPY CIGPOUT TO CIGIN
         FLMTRNSL FUNCTN=BUILD,PORDER=0,CALLNAM='STEP1C: CLZSLR',      X
               COMPILE=CLZSLR,DSNAME=CLZ.SCLZLOAD
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W
         FLMALLOC DDNAME=CIGIN,IOTYPE=U
         FLMALLOC DDNAME=CIGPUNCH,IOTYPE=W,LRECL=300
*                                        COPY CIGPUNCH TO FTPIN
         FLMTRNSL FUNCTN=BUILD,CALLNAM='STEP2D: CLZFCOPY',             X
               COMPILE=CLZFCOPY,DSNAME=CLZ.SCLZLOAD,                   X
               OPTIONS='CIGPUNCH,FTPIN   '
         FLMALLOC DDNAME=CIGPUNCH,IOTYPE=U
         FLMALLOC DDNAME=FTPIN,IOTYPE=W,LRECL=300
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W
*********************************************************************
*
*  STEP 2:  CREATE FTP COMMANDS AND CALL FTP
*
*********************************************************************
         FLMTRNSL FUNCTN=BUILD,CALLNAM='STEP2A: IRXJCL',               X
               COMPILE=IRXJCL,                                         X
               OPTIONS='CLZRFTP1 MEMBER=@@FLMMBR,PROJECT=@@FLMPRJ,GROUPX
               =@@FLMGRP,TYPE=@@FLMTYP'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZJCL
         FLMALLOC DDNAME=SYSTSPRT,IOTYPE=U
         FLMALLOC DDNAME=SYSTSIN,IOTYPE=U
         FLMALLOC DDNAME=FTPIN,IOTYPE=U
         FLMALLOC DDNAME=INPUT,IOTYPE=W,LRECL=300   USED BY FTP
         FLMALLOC DDNAME=OUTPUT,IOTYPE=W            USED BY FTP
```

Figure 55. CLZ@FTP1 S-FTP Translator (Part 2 of 3)

```
*********************************************************************
*
* PROMOTE TRANSLATOR
*
*  STEP 1A: GET THE LONGNAME FROM THE SLR
*
*********************************************************************
         FLMTRNSL FUNCTN=COPY,CALLNAM='STEP1A: CLZFPARM',              X
               PDSDATA=Y,                                              X
               COMPILE=CLZFPARM,DSNAME=CLZ.SCLZLOAD,                   X
               OPTIONS='NEW,LIST LONGNAME WHERE SHORTNAME = @@FLMMBR.'
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W
         FLMALLOC DDNAME=CIGPOUT,IOTYPE=W
*                                        COPY CIGPOUT TO CIGIN
         FLMTRNSL FUNCTN=COPY,CALLNAM='STEP1B: CLZFCOPY',              X
               PDSDATA=Y,                                              X
               COMPILE=CLZFCOPY,DSNAME=CLZ.SCLZLOAD,                   X
               OPTIONS='CIGPOUT ,CIGIN   '
         FLMALLOC DDNAME=CIGPOUT,IOTYPE=U
         FLMALLOC DDNAME=CIGIN,IOTYPE=W
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W,PRINT=Y
*                                        COPY CIGPOUT TO CIGIN
         FLMTRNSL FUNCTN=COPY,PORDER=0,CALLNAM='STEP1C: CLZSLR',       X
               PDSDATA=Y,                                              X
               COMPILE=CLZSLR,DSNAME=CLZ.SCLZLOAD
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W
         FLMALLOC DDNAME=CIGIN,IOTYPE=U
         FLMALLOC DDNAME=CIGPUNCH,IOTYPE=W
*                                        COPY CIGPUNCH TO FTPIN
         FLMTRNSL FUNCTN=COPY,CALLNAM='STEP2D: CLZFCOPY',              X
               PDSDATA=Y,                                              X
               COMPILE=CLZFCOPY,DSNAME=CLZ.SCLZLOAD,                   X
               OPTIONS='CIGPUNCH,FTPIN   '
         FLMALLOC DDNAME=CIGPUNCH,IOTYPE=U
         FLMALLOC DDNAME=FTPIN,IOTYPE=W
         FLMALLOC DDNAME=CIGLOG,IOTYPE=W
*********************************************************************
*
*  STEP 2:  CREATE FTP COMMANDS AND CALL FTP
*
*********************************************************************
         FLMTRNSL FUNCTN=COPY,CALLNAM='STEP2A: IRXJCL',                X
               PDSDATA=Y,                                              X
               COMPILE=IRXJCL,                                         X
               OPTIONS='CLZRFTP1 MEMBER=@@FLMMBR,PROJECT=@@FLMPRJ,GROUPX
               =@@FLMGRP,TYPE=@@FLMTYP'
         FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
           FLMCPYLB CLZ.SCLZJCL
         FLMALLOC DDNAME=SYSTSPRT,IOTYPE=U
         FLMALLOC DDNAME=SYSTSIN,IOTYPE=U
         FLMALLOC DDNAME=FTPIN,IOTYPE=U
         FLMALLOC DDNAME=INPUT,IOTYPE=W            USED BY FTP
         FLMALLOC DDNAME=OUTPUT,IOTYPE=W           USED BY FTP
```

Figure 55. CLZ@FTP1 S-FTP Translator (Part 3 of 3)

Step 4: Review and modify CLZRFTP1 REXX script

In this step, you review and modify the CLZRFTP1 REXX script found in the **CLZ.SCLZJCL** library delivered with the SMP/E installation of Cloud 9.

Note: Issue the CAPS OFF command BEFORE you start to edit this member.

1. Using ISPF EDIT, access member CLZRFTP1 in the CLZ.SCLZJCL library.
2. Issue the CAPS OFF command to ensure case sensitivity.
3. Substitute your site-specific values (identified in Table 36 on page 157)

```
/* rexx */
/*  ------------------------------------ */
/*  CLZRFTP1 - USED WITH CLZ@FTP1 TRANSLATOR*/
/*  ------------------------------------ */
/*  This is a rexx program that invokes FTP */
/*  to ship SCLM    inventory to specified  */
/*  locations out in the network.           */
/*                                          */
/*  This is prototype and must be customized*/
/*  by the user.                            */
/*                                          */
/*  ------------------------------------ */

    TRACE ALL           /* Delete this to eliminate trace */

/*  ------------------------------------- */
/*  The input parameters passed by the caller */
/*  are in the following order:               */
/*                                            */
/*  1:  member=shortname                      */
/*  2:  project=project                       */
/*  3:  group=group                           */
/*  4:  type=type                             */
/*  ------------------------------------- */

    parse arg request
    fx = 0
    true = 1
    false = 0
    ftpIdx=0

    call GetParms
    call GetPCFileName
    call GetTranslationType
/*
   GROUP=DEV
*/
    if (group == 'DEV') then do

       /* -------------------- */
       /* Set Target Location   */
       /* Example of ftp to ISP */
       /* -------------------- */

       /*  ftp -> Surfnet web2.surfnetcorp.com */
        ipaddr='999.999.999.99'; port=21
        user='userid'; password='pass'; dir='/isp/userdir'

       /* ---------------------------------------------------- */
       /* Build FTP commands and invoke FTP.                   */
       /* ---------------------------------------------------- */

        call InvokeFTP
```

Figure 56. CLZRFTP1 REXX Script (Part 1 of 5)

```
      /* ---------------------------------------------------- */
      /* Set Target Location                                  */
      /* Example of ftp to a OS/390 Unix System Services location */
      /* ---------------------------------------------------- */

       /*  ftp -> os/390 */
       ipaddr='999.999.999.99'; port=21
       user='userid'; password='pass'; dir='/u/userdir'

      /* ---------------------------------------------------- */
      /* Build FTP commands and invoke FTP.                   */
      /* ---------------------------------------------------- */

       call InvokeFTP

   end
/*
   GROUP=QA
*/
    if (group == 'QA') then do


      /* ---------------------------------------------------- */
      /* Set Target Location                                  */
      /* Example of ftp to a Windows machine on the network.      */
      /* ---------------------------------------------------- */

       /*  ftp -> CIG demo machine   */
       ipaddr='999.999.999.99'; port=21
       user='userid'; password='pass'; dir='c:\userdir'

      /* ---------------------------------------------------- */
      /* Build FTP commands and invoke FTP.                   */
      /* ---------------------------------------------------- */

       call InvokeFTP


      /* ---------------------------------------------------- */
      /* Set Target Location                                  */
      /* example of ftp to a the A: drive to cut a disk.          */
      /* ---------------------------------------------------- */

      /*  ftp -> Demo Machine A: Drive*/

       ipaddr='999.999.999.99'; port=21
       user='userid'; password='pass'; dir='a:\userdir'

      /* ---------------------------------------------------- */
      /* Build FTP commands and invoke FTP.                   */
      /* ---------------------------------------------------- */

       call InvokeFTP
    end
return
```

Figure 56. CLZRFTP1 REXX Script (Part 2 of 5)

```
/* ---------------------------------------------------------------- */
/*   Get parms                                                      */
/*   There will be four parms.                                      */
/* ---------------------------------------------------------------- */
GetParms:
    do while (request ¬= '')
        parse var request parm','request
        parse var parm var1"="val1
        interpret var1"="'val1'
    end
return

/* ---------------------------------------------------------------- */
/*   Get longname from //FTPIN                                      */
/* ---------------------------------------------------------------- */
GetPCFileName:
    address MVS "EXECIO * DISKR FTPIN (STEM input. FINIS"
    i = input.0 - 1
    pcFileName = input.i
    pcFileName = strip(pcFileName,'B',' ')
/* ---------------------------------- */
/* Create default .exe name for return. */
/* ---------------------------------- */

     if type = 'MAKE' then
     do
      wheredot = lastpos('.',pcFileName)
      pcFileNamexe = substr(pcFileName,1,wheredot)||'exe'
     end

return

/* ---------------------------------------------------------------- */
/*   Get translation type                                           */
/*   The purpose of this routine is to determine the file attribute  */
/*   for the source file of the FTP. This is determined by extension */
/*   type. The first two lines capture the extension for analysis.   */
/*   These lines should not be modified.                            */
/* ---------------------------------------------------------------- */

GetTranslationType:
    fileExtension = substr(pcFileName,lastpos('.',pcFileName)+1)
    fileExtension = translate(fileExtension)    /* upper case */

/* ---------------------------------------------------------------- */
/* Modify the if statements to include your binary types here.       */
/*       GIF    = BINARY                                           */
/*       JPG    = BINARY                                           */
/*       others = ASCII   ( default )                              */
/* ---------------------------------------------------------------- */

    if (fileExtension == 'GIF')       then translationType = 'BINARY'
    else if (fileExtension == 'JPG')  then translationType = 'BINARY'
    else translationType = 'ASCII'

return
```

Figure 56. CLZRFTP1 REXX Script (Part 3 of 5)

```
/* ---------------------------------------------------------------- */
/*   Format FTP commands                                            */
/* ---------------------------------------------------------------- */
InvokeFTP:
    ftpIdx = 0
    ftpIdx=ftpIdx+1; ftp.ftpIdx=ipaddr' 'port
    ftpIdx=ftpIdx+1; ftp.ftpIdx=user
    ftpIdx=ftpIdx+1; ftp.ftpIdx=password
    ftpIdx=ftpIdx+1; ftp.ftpIdx="lcd '"project"."group"."type"'"
    ftpIdx=ftpIdx+1; ftp.ftpIdx="cd "dir

      /* ----------------------------------------  */
      /* sync the source file                     */
      /* ----------------------------------------  */

    ftpIdx=ftpIdx+1; ftp.ftpIdx="put "member" "pcFileName

      /* ----------------------------------------- */
      /* if 'make' type then issue exec MAKE command  */
      /* ----------------------------------------- */

       if (type  == 'MAKE' ) then do

          /* ----------------------------------------  */
          /* execute the make file on the remote machine */
          /* ----------------------------------------  */

          ftpIdx=ftpIdx+1; ftp.ftpIdx="quote site exec " pcFileName

          /* ----------------------------------------  */
          /* reset the host target directory to 'exe'.   */
          /* reset translation type to binary.           */
          /* request a return of the 'exe' to host.      */
          /* ----------------------------------------  */

          ftpIdx=ftpIdx+1; ftp.ftpIdx="lcd '"mbrexe"'"
          ftpIdx=ftpIdx+1; ftp.ftpIdx="binary"

          /* ----------------------------------------  */
          /* assumption: the exec name is same as the    */
          /* .c source.  This will not always be the     */
          /* case.  Per compiler, the rules of output    */
          /* creation and naming standards will need     */
          /* to be examined.                             */
          /* ----------------------------------------  */

          ftpIdx=ftpIdx+1
          ftp.ftpIdx="get "pcFileNamexe member" "mbrexe "replace"

          ftpIdx=ftpIdx+1; ftp.ftpIdx="lcd '"mbrlog"'"
          ftpIdx=ftpIdx+1; ftp.ftpIdx="ASCII"
          ftpIdx=ftpIdx+1

       end

    ftpIdx=ftpIdx+1; ftp.ftpIdx="quit"
    ftp.0 = ftpIdx
```

Figure 56. CLZRFTP1 REXX Script (Part 4 of 5)

```
    /* ------------------------------------------- */
    /* write ftp commands to //input and invoke ftp */
    /* ------------------------------------------- */

    address MVS "EXECIO * DISKW INPUT (STEM ftp. FINIS"

    /* ------------------------------------------- */
    /* Attach FTP and execute commands.            */
    /* ------------------------------------------- */

    address attach FTP       /* here we invoke FTP */

    /* --------------------------- */
    /* read ftp output into an array */
    /* --------------------------- */

    address MVS "EXECIO * DISKR OUTPUT (STEM input. FINIS"

    /* --------------------------- */
    /* copy to a user dataset       */
    /* --------------------------- */

    "alloc dd(MSGFILE) mod reuse da('"USERID()".FTPLOG')"
    "EXECIO * DISKW MSGFILE (STEM input. FINIS"

    "free dd(MSGFILE)"
return
```

Figure 56. CLZRFTP1 REXX Script (Part 5 of 5)

Chapter 18. Create SCLM and Cloud 9 definitions

Step 5. Update your Project Definition

The purpose of this step is to review the requirements for updating your current project definition to include the new CLZ@FTP1 translator. The translator is included in the CLZ.SCLZJCL library. Include this library in your project definition JCL SYSLIB DDNAME so that the compiler can find the language definition member. Figure 57 shows the statements required to include the new translator. Include these statements in your current project definition JCL and submit.

```
*********************************************************************
*                LANGUAGE DEFINITION TABLES
*********************************************************************
*
*
        COPY  CLZ@FTP1           -- FTP BUILD/DEPLOY          --
```

Figure 57. Copy Statement Example

Checkpoint #2 for S-FTP installation

At this point you should have completed the following tasks.

Table 39. Checkpoint #2 for S-FTP Installation

Task	Completed?
Reviewed and modified translator CLZ@FTP1	
Reviewed and modified REXX script CLZRFTP1	
Updated your project definition include the CLZ@FTP1	

Chapter 19. Test the S-FTP translator

Step 6. Add an HTML file

To test the S-FTP Translator, perform the following steps:

1. Add a piece of source code defined to the type and language supported. For example, add a type called HTML with a language of FTP1. For information about adding source to SCLM libraries, see the *IBM Cloud 9 for SCLM for z/OS User's Guide*.
2. Through Cloud 9, request a build of the source to invoke the **Build CLZ@FTP1** translator. View the batch job output. Figure 58 is an example of the BLDMSGS output.

```
********************* TOP OF DATA ***************************
FLM42000 - BUILD PROCESSOR INITIATED - 16:52:33 ON 01/04/30

FLM44500 - >> INVOKING BUILD TRANSLATOR(S) FOR TYPE: HTML MEMBER: WE$SH001
FLM06501 - TRANSLATOR RETURN CODE FROM ===> STEP1A: CIGFPARM     ===> 0
FLM06501 - TRANSLATOR RETURN CODE FROM ===> STEP1B: CIGFCOPY     ===> 0
FLM06501 - TRANSLATOR RETURN CODE FROM ===> STEP1C: C9LSLR       ===> 0
FLM06501 - TRANSLATOR RETURN CODE FROM ===> STEP2D: CIGFCOPY     ===> 0
FLM06501 - TRANSLATOR RETURN CODE FROM ===> STEP2A: IRXJCL       ===> 0
FLM46000 - BUILD PROCESSOR COMPLETED - 16:53:03 ON 01/04/30
******************** BOTTOM OF DATA ***************************
```

Figure 58. BLDMSGS Output

3. Figure 59 on page 174 shows an example of the output produced by the S-FTP REXX exec because it has the "trace all" command coded into it. It shows you the progress of the FTP process and where problem areas (if any) have occurred. Errors appear as non-zero return codes. Use this output in conjunction with the log in Figure 60 on page 176 to follow the complete process.

```
READY
  ISPSTART CMD(%TEMPNAME)
    3 *-* /*  ---------------------------------------- */
    4 *-* /*  CLZRFTP1 - USED WITH CLZ@FTP1 TRANSLATOR  */
    5 *-* /*  ---------------------------------------- */
    6 *-* /*  This is a rexx program that invokes FTP   */
    7 *-* /*  to ship SCLM    inventory to specified    */
    8 *-* /*  locations out in the network.             */
    9 *-* /*                                           */
   10 *-* /*  This is prototype and must be customized  */
   11 *-* /*  by the user.                              */
   12 *-* /*                                           */
   13 *-* /* ----------------------------------------   */
   16 *-* /* ----------------------------------------  */
   17 *-* /* The input parameters passed by the caller  */
   18 *-* /* are in the following order:                */
   19 *-* /*                                           */
   20 *-* /*  1:  member=shortname                      */
   21 *-* /*  2:  project=project                      */
   22 *-* /*  3:  group=group                          */
   23 *-* /*  4:  type=type                            */
   24 *-* /* ---------------------------------------- */
   26 *-* parse arg request
   27 *-* fx = 0
   28 *-* true = 1
   29 *-* false = 0
   30 *-* ftpIdx=0
   32 *-* call GetParms
  100 *-*  GetParms:
  101 *-*  do while (request = '')
  102 *-*   parse var request parm','request
  103 *-*   parse var parm var1"="val1
  104 *-*   interpret var1"="'val1'
      *-*    MEMBER=val1

(More..)

   38 *-* if (group == 'DEV1')
     *-*  then
     *-*  do
  40 *-*   /* ------------------------------------------------ */
  41 *-*   /* Set Target Location                              */
  42 *-*   /* Example of ftp to a z/OS Unix System Services location   */
  43 *-*   /* ------------------------------------------------------- */
  45 *-*   /*  ftp -> z/OS   */
```

Figure 59. REXX Script Trace Data (Part 1 of 2)

```
46 *-*   ipaddr='999.99.999.99'
    *-*   port=21
 47 *-*   user='?????'
    *-*   password='?????'
    *-*   dir='/u/cig8002/c9demo'
 49 *-*   /* ------------------------------------------------- */
 50 *-*   /* Build FTP commands and invoke FTP.                */
 51 *-*   /* ------------------------------------------------- */
 53 *-*   call InvokeFTP
157 *-*    InvokeFTP:
158 *-*    ftpIdx = 0
 159 *-*    ftpIdx=ftpIdx+1
     *-*    ftp.ftpIdx=ipaddr' 'port
 160 *-*    ftpIdx=ftpIdx+1
     *-*    ftp.ftpIdx=user
 161 *-*    ftpIdx=ftpIdx+1
     *-*    ftp.ftpIdx=password
 162 *-*    ftpIdx=ftpIdx+1
     *-*    ftp.ftpIdx="lcd '"project"."group"."type"'"
 163 *-*    ftpIdx=ftpIdx+1
     *-*    ftp.ftpIdx="cd "dir
 165 *-*    /* ----------------------------------------  */
 166 *-*    /* sync the source file                      */
 167 *-*    /* ----------------------------------------  */
 169 *-*    ftpIdx=ftpIdx+1
     *-*    ftp.ftpIdx="put "member" "pcFileName
 171 *-*    /* ------------------------------------------- */
 172 *-*    /* if 'make' type then issue exec MAKE command  */
 173 *-*    /* ------------------------------------------- */
 175 *-*    if (type  == 'MAKE' )
 209 *-*    ftpIdx=ftpIdx+1
     *-*    ftp.ftpIdx="quit"
 210 *-*    ftp.0 = ftpIdx
 212 *-*    /* ------------------------------------------- */
 213 *-*    /* write ftp commands to //input and invoke ftp */
 214 *-*    /* ------------------------------------------- */
 216 *-*    address MVS "EXECIO * DISKW INPUT (STEM ftp. FINIS"
     >>>      "EXECIO * DISKW INPUT (STEM ftp. FINIS"
 218 *-*    /* ------------------------------------------- */
 219 *-*    /* Attach FTP and execute commands.             */
 220 *-*    /* ------------------------------------------- */
 222 *-*    address attach FTP       /* here we invoke FTP */
     >>>      "FTP"
 224 *-*    /* ---------------------------- */
 225 *-*    /* read ftp output into an array */
 226 *-*    /* ---------------------------- */
 228 *-*    address MVS "EXECIO * DISKR OUTPUT (STEM input. FINIS"
     >>>      "EXECIO * DISKR OUTPUT (STEM input. FINIS"
 230 *-*    /* ---------------------------- */
 231 *-*    /* copy to a user dataset        */
 232 *-*    /* ---------------------------- */
 234 *-*    "alloc dd(MSGFILE) mod reuse da('CIGT.FULL.RBLOG')"
     >>>      "alloc dd(MSGFILE) mod reuse da('CIGT.FULL.RBLOG')"
 235 *-*    "EXECIO * DISKW MSGFILE (STEM input. FINIS"
     >>>      "EXECIO * DISKW MSGFILE (STEM input. FINIS"
 237 *-*    "free dd(MSGFILE)"
     >>>      "free dd(MSGFILE)"
 238 *-*    return
  55 *-*  end
```

Figure 59. REXX Script Trace Data (Part 2 of 2)

4. View the 'userid.ftplog' data set updated in the CLZRFTP1 script. There should be data in the file and it should be similar to that in Figure 60 on page 176.

Any real ID's or IP addresses have been changed to dummy values for security purposes.

```
 EZA1736I FTP
 EZA1450I IBM FTP CS V2R7  1998 282 22:42 UTC
 EZA1466I FTP: using TCPIP
 EZA1456I Connect to ?
 EZA1736I 999.99.999.99 21
 EZA1554I Connecting to:  999.99.999.99 port: 21.
 220-FTPD1 IBM FTP CS V2R7 at P390, 22:04:50 on 2001-04-30.
 220 Connection will close if idle for more than 20 minutes.
 EZA1459I NAME (999.999.999.99:XXXXX):
 EZA1701I >>> USER XXXXX
 331 Send password please.
EZA1789I PASSWORD:
EZA1701I >>> PASS
230 P390C is logged on.  Working directory is "XXXXX.".
EZA1460I Command:
EZA1736I lcd 'CIGDEMO.DEV1.HTML'
EZA2081I Local directory name set to partitioned data set CIGDEMO.DEV1.
EZA1460I Command:
EZA1736I cd /u/cig8002/c9demo
EZA1701I >>> CWD /u/cig8002/c9demo
250 HFS directory /u/cig8002/c9demo is the current working directory
EZA1460I Command:
EZA1736I put WE$SHOUL  'We should ship the web cast this way!.html'
EZA1701I >>> SITE VARrecfm LRECL=256 RECFM=VB BLKSIZE=2564
200 Site command was accepted
EZA1701I >>> PORT 999,99,999,99,4,12
200 Port request OK.
EZA1701I >>> STOR  'We should ship the web cast this way!.html'
125 Storing data set /u/cig8002/c9demo/ We should ship the web cast this
250 Transfer completed successfully.
EZA1617I 52255 bytes transferred in 0.240 seconds.  Transfer rate 217.73
EZA1460I Command:
EZA1736I quit
EZA1701I >>> QUIT
```

Figure 60. User FTP Log Example

Part 5. Cloud 9 VisualAge for Java Plug-in

Chapter 20. Visual Age for Java Plug-in installation overview

This part of the manual contains the installation procedure for the IBM Cloud 9 Visual Age for Java Plug-in. This is a plug-in that your users can install onto their PCs, so that they can use the Version Control features of VA Java. The installation is performed using InstallShield. The setup files and help HTML for the VA Java Plug-in are all stored in UNIX System Services in a directory beneath the Cloud 9 directory in which the base product was installed. Hereinafter, the following names are used in this manual:

- IBM Cloud 9 for SCLM for z/OS is called Cloud 9
- UNIX System Services and HFS is called USS
- IBM Cloud 9 Visual Age for Java Plug-in is called the VA Java Plug-in
- IBM Breeze for SCLM for z/OS is called Breeze

Product components used during installation

The following sample HTML member can be modified during the installation process. It is located in SCLZHTML and, as a result of the SMP/E installation, has been copied into the root directory where Cloud 9 is installed. The names are provided here as an overview of naming standards and component functionality.

HTML members modified during installation

CLZC9IDX
Cloud 9 index page. Called C9index.htm in USS.

CLZVACHM
Input to the second step of job CLZJVAHI.

CLZJVAHI
Copies the VA Java Plug-in code and VA Java help files from the SCLZHTML library, where SMP/E has installed them, into USS.

A step-by-step approach

This section provides an overview of the steps involved in installing the Cloud 9 VA Java Plug-in Interface.

Table 40. VA Java Plug-in Installation Steps

Before you begin	
1.	Review system and software considerations
Create and populate HFS Cloud 9 directories for VA Java Plug-in	
2.	Modify CLZVACHM
3.	Modify and submit CLZJVAHI
HTML Tailoring	
4.	Tailor C9index.htm
Perform the Installation of the VA Java Plug-in	
5.	Execute C9index.htm
6.	Run InstallShield

Visual Age for Java Plug-in installation overview

Chapter 21. Before you begin

This section describes the preparation steps that you undertake before starting the installation of the Cloud 9 VisualAge for Java Plug-in.

Step 1: Review software and hardware requirements

In this step, you review the system, software and hardware requirements for product installation.

System requirements

The VisualAge for Java Plug-in has been tested using Visual Age for Java Enterprise Editions 3.5.3 and 4.0. Older versions of the product might work but have not been tested.

The VA Java Plug-in has been tested on the following platforms:

- NT 4.0 SP5 (no longer supported by Microsoft)
- NT 4.0 SP6 (free upgrade)
- Windows 2000 Professional
- Windows 2000 Professional SP1
- Windows 2000 Professional SP2
- Windows 2000 Professional SP3
- Windows XP Professional

SMP/E UNIX considerations

During the Cloud 9 SMP/E installion, UNIX directories, based on a ***PathPrefix*** variable, were created and populated. If the default values are used by your installation, your rootdir value is equal to:

```
/usr/lpp/Cloud9/
```

Your system administrator can provide more information.

Chapter 22. Create and populate HFS Cloud 9 directory for VA Java Plug-in

In these steps, you create the Cloud 9 Unix directory that contains the VA Java Plug-in and VA Java help files. Because many of the members contain case-sensitive values, please issue the CAPS OFF command to ensure that automatic upper casing does not occur.

Step 2: Modify CLZVACHM

The REXX exec CLZVACHM is input to the second step of job CLZJVAHI. This REXX exec performs the CHMOD commands against the imported files to correctly set the permissions.

To modify CLZVACHM, perform the following tasks:

1. Using ISPF EDIT, access member CLZVACHM in the CLZ.SCLZJCL target library.
2. Issue the CAPS OFF command to ensure that case sensitive values are not converted to upper case.

   WARNING: Unix files are case sensitive. Do not change the case on any file names contained in this REXX EXEC.
3. Substitute your site-specific values (identified in the "Step 2: Record site–specific information" on page 6), as instructed in the comment area of the member.
4. Save the member.

Step 3: Modify and Submit CLZJVAHI

Job CLZJVAHI copies the VA Java Plug-in code and VA Java help files from the SCLZHTML library, where SMP/E has installed them, into the required directory on your USS server.

To modify CLZVACHM, perform the following tasks:

1. Using ISPF EDIT, access member CLZJVAHI in the CLZ.SCLZJCL target library.
2. Issue the CAPS OFF command to ensure that case sensitive values are not converted to upper case.

   WARNING: Unix files are case sensitive. Do not change the names contained in this JCL.
3. Copy your job card values to the top of the member.
4. Substitute your site-specific values (identified in the "Step 2: Record site–specific information" on page 6), as instructed in the comment area of the member.
5. Save the member.
6. Submit the job.

Note: This job should terminate with COND CODE=0. If it does not:

1. Review your job card parameters and the JCL for errors.

2. Resubmit the job.

After this job has been submitted and successfully executed, the Cloud 9 VA Java USS directory should be populated and ready to use to download the VA Java Plug-in and to browse the VA Java help file.

Chapter 23. HTML tailoring

In order to enable users to install the VA Java Plug-in onto their PC's, an InstallShield setup file has been installed by SMP/E into USS. To access this setup file, a sample index page has been provided that can be tailored and used by the installer.

This index page provides links to the installation programs and the online help. It also provides links to the Web-based SCLM Suite products, along with the SCLM Suite documentation link pages. Throughout the installation process, it has been assumed that you are using the C9index.htm included with the product. If you choose to tailor your own index page, substitute your page name for C9index.htm thoughout this installation process.

Step 4: Tailor C9index.htm

In this step, you tailor the sample index HTML that your users access to install the plug-in. As the index is installed in the same location as the Cloud 9 home page, you invoke it in the same way. All links on the page are relative to this home directory, so very little tailoring is required. The only address in the index page that must be tailored is the Breeze Web browser html. If you have not installed Breeze at your site, you do not need to tailor this and you can remove it from the sample index.

To tailor the C9index.htm, perform the following tasks:

1. Using the OpenMVS ISPF Shell (TSO ISHELL from the command line), enter root directory where you installed Cloud 9. If the default values were used, this is `/usr/lpp/Cloud9`.
2. In the resulting list, find C9index.htm and edit it, by placing an ″e″ next to the file name.
3. If you have installed Breeze at your site, substitute your site-specific values for *ip-address* and *portno*, for the location of the Breeze Server.
4. Review the contents of the C9index.htm file and save it.

HTML tailoring

Chapter 24. Perform the installation of the VA Java Plug-in

In these steps, you perform the actual InstallShield installation of the VA Java Plug-in. At this point it has been assumed that you have completed the installation of the Cloud 9 base product, that you have successfully logged onto the Cloud 9 system and that you are familiar with the login aspects of the HTTP server.

Step 5: Execute C9index.htm

To invoke the sample index page, execute the C9index.htm file as follows:

1. On your desktop, start your browser.
2. Modify the following statement with your IP address and port number, then type it in the browser's address window (labeled "Location" in Netscape Navigator and "Address" in Internet Explorer):

   ```
   http://ip-address:portno/C9index.htm
   ```

   The browser requests the html file directly from HTTP and executes the Cloud 9 index HTML.

   Note: Ensure that you enter the address using a capital "C". The file is stored in USS and, therefore, the names are case sensitive.
3. Before the index page is displayed, you are prompted with a log-in window. Enter your TSO user ID and password and click **OK** to display the index page.
4. After you have successfully logged in, you see the index page, as shown in Figure 61 on page 188:

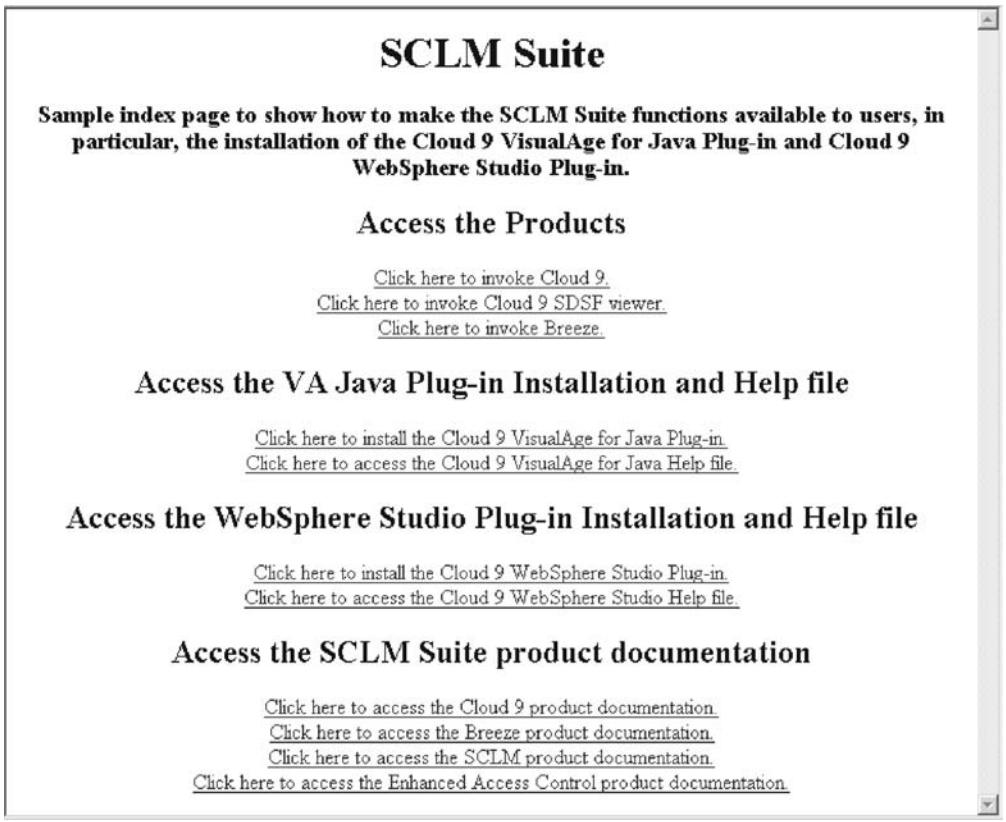

Figure 61. C9index.htm

5. There are two options relating to the VA Java Plug-in. The first is the installation and the second is the online help for the plug-in. At this time, click the link to install the VA Java Plug-in.

   If you are using Internet Explorer, the Save or Run Setup page is displayed:

Figure 62. Save or Run page in Internet Explorer

   If you are using Netscape Navigator, the Save Setup page is displayed:

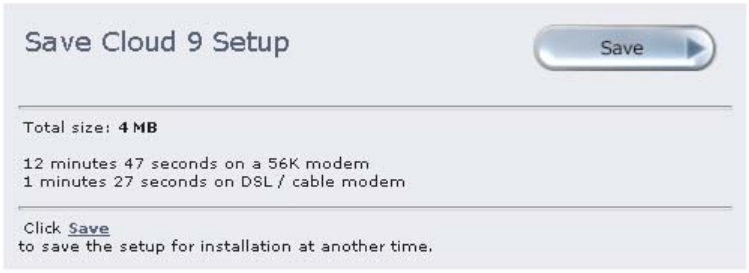

Figure 63. Save or Run page in Netscape

Step 6: Run the InstallShield

The InstallShield installation is slightly different between Internet Explorer and Netscape. Follow the guidelines provided for your browser.

Internet Explorer installation

Depending on your version of Internet Explorer and the settings used, you might be given a choice between running the installation program from its location on the mainframe or downloading the setup.exe to your PC and running it from there, or you might only have the option to download and run the setup.exe.

1. On the Save or Run Setup page, click **Save or Run**.
2. In the File Download dialog box, select from your available choices.
3. If you choose to open the file from it's current location, it immediatly starts the installation process, using the code installed in USS. Follow the installation instuctions in the subsequent InstallShield dialog boxes.
4. If you choose to save the file to disk, Windows asks for a location. Select a location on your hard drive, then use Windows Explorer find the setup.exe file you have just downloaded. Double-click the file and the InstallShield program runs. Follow the instructions in the subsequent InstallShield dialog boxes.

Netscape Navigator installation

With Netscape, you cannot run the setup.exe directly from USS, therefore, you must download it to your PC and run it from there.

1. On the Save Setup page, click **Save**.
2. In the Unknown File Type dialog box, click **Save**.
3. Select a location on your hard drive, then use Windows Explorer find the setup.exe file you have just downloaded. Double-click on the file and the InstallShield program runs. Follow the instructions in the subsequent InstallShield dialog boxes.

Perform the installation of the VA Java Plug-in

Part 6. Cloud 9 WebSphere Studio Application Developer Plug-in

Chapter 25. Cloud 9 WebSphere Studio Application Developer Plug-In installation overview

This part of the manual contains the installation procedure for the IBM Cloud 9 for SCLM for z/OS WebSphere Studio Application Developer Plug-In (WSAD Plug-in). This is a plug-in that your users can install onto their PCs, so that they can use the Version Control features of WebSphere Studio Application Developer or WebSphere Studio Enterprise Developer. Because the WSAD Plug-in is configured and works in exactly the same way with either WebSphere product, this manual uses the name "WebSphere Studio Application Developer" to refer to both WSAD and WSED.

The installation is performed using InstallShield. The setup files and help HTML for the Cloud 9 WebSphere Studio Application Developer interface are all stored in UNIX System Services in a directory beneath the Cloud 9 directory in which the base product was installed. Hereinafter, the following names are used in this manual:

- IBM Cloud 9 for SCLM for z/OS is called Cloud 9
- UNIX System Services and HFS is called USS
- IBM Cloud 9 for SCLM for z/OS WebSphere Studio Application Developer Plug-In is called the WSAD Plug-In
- IBM Breeze for SCLM for z/OS is called Breeze

Product components used during installation

The following sample HTML member can be modified during the installation process. It is located in SCLZHTML and, as a result of the SMP/E installation, has been copied into the root directory where Cloud 9 is installed. The names are provided here as an overview of naming standards and component functionality.

HTML members modified during installation

CLZC9IDX
Cloud 9 index page. Called C9index.htm in USS.

CLZWACHM
Input to the second step of job CLZWACHM.

CLZWACHM
Copies the WSAD Plug-in code and WSAD help files from the SCLZHTML library, where SMP/E has installed them, into USS.

A step-by-step approach

This section provides an overview of the steps involved in installing the Cloud 9 WebSphere Studio Application Developer Plug-in.

Table 41. WSAD Plug-in Installation Steps

Before you begin	
1.	Review system and software considerations
Create and populate HFS Cloud 9 directories for WSAD Plug-in	

Table 41. WSAD Plug-in Installation Steps (continued)

2.	Modify CLZWACHM
3.	Modify and submit CLZJWAHI
HTML Tailoring	
4.	Tailor C9index.htm
Perform the Installation of the WSAD Plug-in	
5.	Execute C9index.htm
6.	Run InstallShield

Chapter 26. Before you begin

This section describes the preparation steps that you undertake before starting the installation of the Cloud 9 WSAD Plug-in.

Step 1: Review software and hardware requirements

In this step, you review the system, software and hardware requirements for product installation.

System requirements

In addition to the standard installation of Cloud 9, the following system requirements must be in place to install the WSAD Plug-in.

The WSAD Plug-in has been tested using WebSphere Studio Application Developer Edition 5.0 and WebSphere Studio Enterprise Developer Edition 5.0. Older versions of the product might work but have not, as of yet, been tested.

The WSAD Plug-in has been tested on the following platforms:

- NT 4.0 SP5 (no longer supported by Microsoft)
- NT 4.0 SP6 (free upgrade)
- Windows 2000 Professional
- Windows 2000 Professional SP1
- Windows 2000 Professional SP2
- Windows 2000 Professional SP3
- Windows XP Professional

SMP/E UNIX considerations

During the Cloud 9 SMP/E installion, UNIX directories, based on a ***PathPrefix*** variable, were created and populated. If the default values are used by your installation, your rootdir value is equal to:

```
/usr/lpp/Cloud9/
```

Your system administrator can provide more information.

Chapter 27. Create and populate HFS Cloud 9 directory for WSAD Plug-in

In these steps, you create the Cloud 9 Unix directory that contains the WSAD Plug-in and WSAD help files. Because many of the members contain case-sensitive values, please issue the CAPS OFF command to ensure that automatic upper casing does not occur.

Step 2: Modify CLZWACHM

The REXX exec CLZWACHM is input to the second step of job CLZJVAHI. This REXX exec performs the CHMOD commands against the imported files to correctly set the permissions.

To modify CLZWACHM, perform the following tasks:

1. Using ISPF EDIT, access member CLZWACHM in the CLZ.SCLZJCL target library.
2. Issue the CAPS OFF command to ensure that case sensitive values are not converted to upper case.

   WARNING: Unix files are case sensitive. Do not change the case on any file names contained in this REXX EXEC.
3. Substitute your site-specific values (identified in the "Step 2: Record site–specific information" on page 6), as instructed in the comment area of the member.
4. Save the member.

Step 3: Modify and Submit CLZJWAHI

Job CLZJWAHI copies the WSAD Plug-in code and WSAD help files from the SCLZHTML library, where SMP/E has installed them, into the required directory on your USS server.

To modify CLZWACHM, perform the following tasks:

1. Using ISPF EDIT, access member CLZJWAHI in the CLZ.SCLZJCL target library.
2. Issue the CAPS OFF command to ensure that case sensitive values are not converted to upper case.

   WARNING: Unix files are case sensitive. Do not change the names contained in this JCL.
3. Copy your job card values to the top of the member.
4. Substitute your site-specific values (identified in the "Step 2: Record site–specific information" on page 6), as instructed in the comment area of the member.
5. Save the member.
6. Submit the job.

Note: This job should terminate with COND CODE=0. If it does not:

1. Review your job card parameters and the JCL for errors.

2. Resubmit the job.

After this job has been submitted and successfully executed, the Cloud 9 WSAD USS directory should be populated and ready to use to download the WSAD Plug-in and to browse the WSAD help file.

Chapter 28. HTML tailoring

In order to enable users to install the WSAD Plug-in onto their PC's, an InstallShield setup file has been installed by SMP/E into USS. To access this setup file, a sample index page has been provided that can be tailored and used by the installer.

This index page provides links to the installation programs and the online help. It also provides links to the Web-based SCLM Suite products, along with the SCLM Suite documentation link pages. Throughout the installation process, it has been assumed that you are using the C9index.htm included with the product. If you choose to tailor your own index page, substitute your page name for C9index.htm thoughout this installation process.

Step 4: Tailor C9index.htm

In this step, you tailor the sample index HTML that your users access to install the plug-in. As the index is installed in the same location as the Cloud 9 home page, you invoke it in the same way. All links on the page are relative to this home directory, so very little tailoring is required. The only address in the index page that must be tailored is the Breeze Web browser html. If you have not installed Breeze at your site, you do not need to tailor this and you can remove it from the sample index.

To tailor the C9index.htm, perform the following tasks:

1. Using the OpenMVS ISPF Shell (TSO ISHELL from the command line), enter root directory where you installed Cloud 9. If the default values were used, this is `/usr/lpp/Cloud9`.
2. In the resulting list, find C9index.htm and edit it, by placing an ″e″ next to the file name.
3. If you have installed Breeze at your site, substitute your site-specific values for *ip-address* and *portno*, for the location of the Breeze Server.
4. Review the contents of the C9index.htm file and save it.

Chapter 29. Perform the installation of the WSAD Plug-in

In these steps, you perform the actual InstallShield installation of the WSAD Plug-in. At this point it has been assumed that you have completed the installation of the Cloud 9 base product, that you have successfully logged onto the Cloud 9 system and that you are familiar with the login aspects of the HTTP server.

Step 5: Execute C9index.htm

To invoke the sample index page, execute the C9index.htm file as follows:

1. On your desktop, start your browser.
2. Modify the following statement with your IP address and port number, then type it in the browser's address window (labeled "Location" in Netscape Navigator and "Address" in Internet Explorer):

   ```
   http://ip-address:portno/C9index.htm
   ```

   The browser requests the html file directly from HTTP and executes the Cloud 9 index HTML.

   Note: Ensure that you enter the address using a capital "C". The file is stored in USS and, therefore, the names are case sensitive.
3. Before the index page is displayed, you are prompted with a log-in window. Enter your TSO user ID and password and click **OK** to display the index page.
4. After you have successfully logged in, you see the index page, as shown in Figure 61 on page 188:

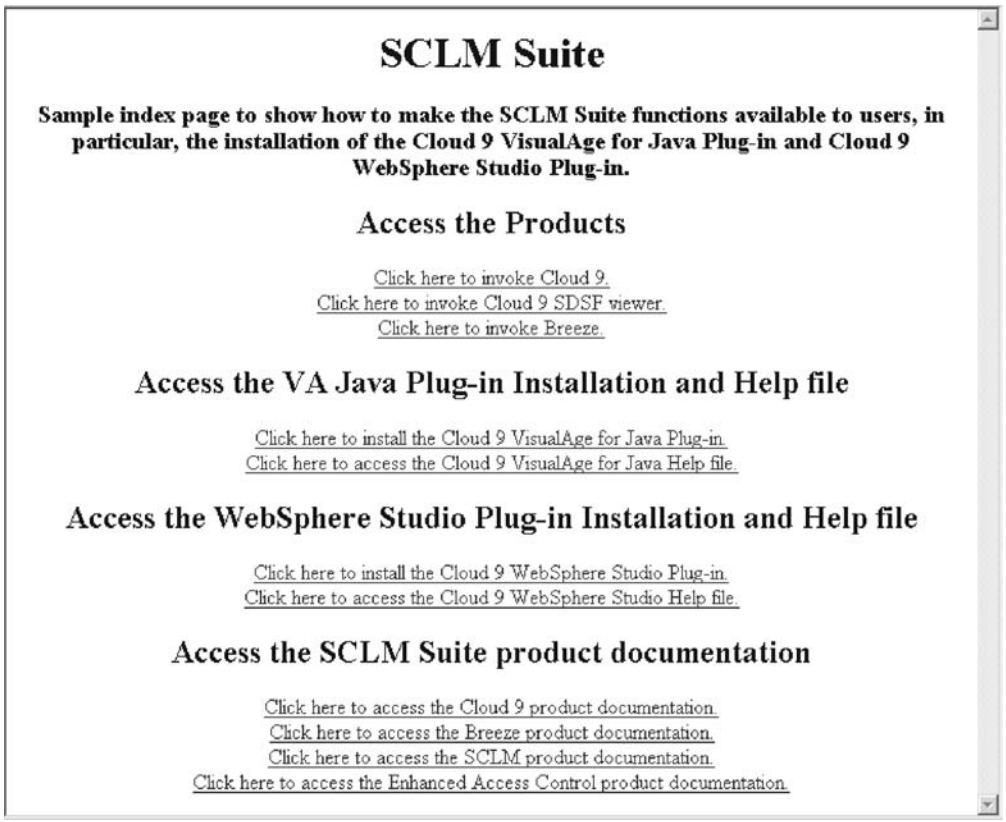

Figure 64. C9index.htm

5. There are two options relating to the WSAD Plug-in. The first is the installation and the second is the online help for the WSAD Plug-in. At this time, click the link to install the Cloud 9 WSAD Plug-in.

   If you are using Internet Explorer, the Save or Run Setup page is displayed:

Figure 65. Save or Run page in Internet Explorer

   If you are using Netscape Navigator, the Save Setup page is displayed:

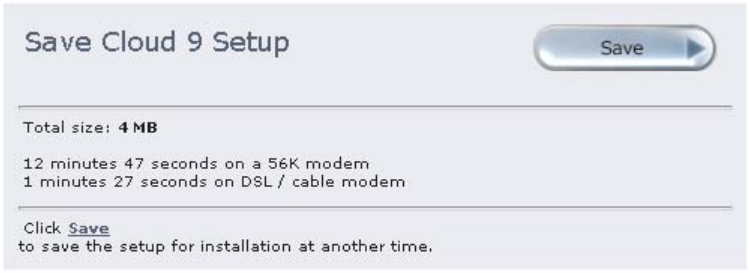

Figure 66. Save or Run page in Netscape

Step 6: Run the InstallShield

The InstallShield installation is slightly different between Internet Explorer and Netscape. Follow the guidelines provided for your browser.

Internet Explorer installation

Depending on your version of Internet Explorer and the settings used, you might be given a choice between running the installation program from its location on the mainframe or downloading the setup.exe to your PC and running it from there, or you might only have the option to download and run the setup.exe.

1. On the Save or Run Setup page, click **Save or Run**.
2. In the File Download dialog box, select from your available choices.
3. If you choose to open the file from it's current location, it immediatly starts the installation process, using the code installed in USS. Follow the installation instuctions in the subsequent InstallShield dialog boxes.
4. If you choose to save the file to disk, Windows asks for a location. Select a location on your hard drive, then use Windows Explorer find the setup.exe file you have just downloaded. Double-click the file and the InstallShield program runs. Follow the instructions in the subsequent InstallShield dialog boxes.

Netscape Navigator installation

With Netscape, you cannot run the setup.exe directly from USS, therefore, you must download it to your PC and run it from there.

1. On the Save Setup page, click **Save**.
2. In the Unknown File Type dialog box, click **Save**.
3. Select a location on your hard drive, then use Windows Explorer find the setup.exe file you have just downloaded. Double-click on the file and the InstallShield program runs. Follow the instructions in the subsequent InstallShield dialog boxes.

Part 7. Appendixes

Appendix A. Cloud 9 UNIX directory structure

The following charts represent the UNIX directory structure and files expected by the Cloud 9 application. The ***rootdir*** value is site specific, all other directory names and file names are not. This includes the case of the file names.

Level 1 — Cloud 9 'rootdir'

/rootdir/

```
   Type  Perm  Changed-EST5EDT   Owner            Size  Filename

_ Dir    755  2003-11-05 04:01  CLOUD9           8192  .
_ Dir    755  2003-10-08 02:19  IBMUSER          8192  ..
_ Dir    755  2003-05-13 02:44  CLOUD9           8192  cgi-bin
_ Dir    755  2003-06-26 01:22  CLOUD9           8192  cloud9
_ File   755  2003-05-11 21:59  CLOUD9            377  cloud9.htm
_ File   777  2003-09-23 22:39  CLOUD9           1594  C9index.htm
_ File   755  2003-12-08 22:12  CLOUD9          16243  httpd.conf
_ File   755  2003-07-21 01:33  CLOUD9           2092  httpd.envvars
_ File   777  2003-05-11 22:08  CLOUD9           5101  httpd.mvsds
_ File   644  2003-12-14 20:02  CLOUD9              9  httpd-pid
_ Dir    755  2003-05-09 05:03  CLOUD9           8192  IBM
_ Dir    777  2003-12-15 00:00  CLOUD9          65536  logs
_ Dir    777  2003-12-08 21:44  CLOUD9           8192  projects
_ Dir    777  2003-05-11 22:31  CLOUD9           8192  reports
_ File   777  2003-05-11 21:59  CLOUD9            420  sdsf.htm
```

Level 2 — cgi-bin directory

/rootdir/cgi-bin/

```
EUID=165   /usr/lpp/products/Cloud9T2/cgi-bin/
   Type  Perm  Changed-EST5EDT   Owner            Size  Filename

_ Dir    755  2003-05-13 02:44  CLOUD9           8192  .
_ Dir    755  2003-11-05 04:01  CLOUD9           8192  ..
_ File   755  2003-05-11 21:54  CLOUD9            940  CLZLABOU
_ File   755  2003-05-11 21:54  CLOUD9            321  CLZRADDS
_ File   755  2003-05-11 21:54  CLOUD9              0  CLZREDRV
_ File   755  2003-05-11 21:54  CLOUD9            322  CLZRENDV
_ File   755  2003-05-11 21:54  CLOUD9           3844  CLZREXIT
_ File   755  2003-05-11 21:54  CLOUD9            838  CLZREX00
_ File   755  2003-05-11 21:54  CLOUD9            331  CLZRINDX
_ File   755  2003-05-11 21:54  CLOUD9            321  CLZRLMBR
_ File   755  2003-05-11 21:54  CLOUD9            322  CLZRLUNX
_ File   755  2003-05-11 21:54  CLOUD9            330  CLZRMENU
_ File   755  2003-05-11 21:54  CLOUD9            322  CLZRMLST
_ File   755  2003-05-11 21:54  CLOUD9            321  CLZRPROF
_ File   755  2003-05-11 21:54  CLOUD9            321  CLZRSCLD
_ File   755  2003-05-11 21:54  CLOUD9            321  CLZRSCLM
_ File   755  2003-05-11 21:54  CLOUD9            322  CLZRSCMA
_ File   755  2003-05-11 21:54  CLOUD9              0  CLZRSDRV
_ File   755  2003-05-11 21:54  CLOUD9            115  CLZRSDSF
_ File   755  2003-05-11 21:54  CLOUD9            115  CLZRSDSM
_ File   755  2003-10-28 22:14  CLOUD9            308  CLZRSMAN
_ File   755  2003-05-11 21:54  CLOUD9            391  CLZRSPDA
_ File   755  2003-05-11 21:54  CLOUD9            322  CLZRULST
```

Level 2 — cloud9 directory

/rootdir/cloud9/

Select one or more files with / or action codes.

```
EUID=165   /usr/lpp/products/Cloud9T2/cloud9/
   Type  Perm  Changed-EST5EDT   Owner            Size  Filename

   Dir    755  03/18/2004 06:48  CLOUD9           8192  .
 _ Dir    775  03/18/2004 08:21  CLOUD9           8192  ..
 _ Dir    777  11/03/2003 08:53  CLOUD9           8192  cache
 _ File   777  10/19/2001 01:29  CLOUD9            273  CLZHMENU.htm
 _ File   777  10/19/2001 00:19  CLOUD9             99  CLZHSDSB.htm
 _ File   777  10/19/2001 01:29  CLOUD9            273  CLZHSDSM.htm
 _ File   777  10/19/2001 01:29  CLOUD9            476  CLZHSDSS.htm
 _ File   777  02/25/2002 00:45  CLOUD9            469  CLZHSPLA.htm
 _ Dir    777  03/19/2003 09:38  CLOUD9           8192  jcl
 _ Dir    777  11/04/2003 01:19  CLOUD9           8192  manifest
 _ Dir    777  03/05/2004 15:34  CLOUD9           8192  profiles
 _ Dir    775  03/18/2004 06:42  CLOUD9           8192  vaj
 _ Dir    775  03/18/2004 06:48  CLOUD9           8192  wsad
```

Level 3 — profiles directory

/rootdir/cloud9/profiles/

Select one or more files with / or action codes

```
  Type   Perm  Changed    (GMT)     Owner      Size    File

 _ Dir    775  10/11/2000 21:28     TCPIP      8192    .
 _ Dir    775  10/11/2000 21:25     TCPIP      8192    ..
 _ File   755  09/28/2000 21:27     TCPIP     12133    P390C.jpg
 _ File   755  09/28/2000 22:20     TCPIP       225    P390C.prf
 _ File   600  09/28/2000 21:27     TCPIP     19529    P390K.jpg
 _ File   755  10/05/2000 21:30     TCPIP       169    P390K.prf
```

Note: The profiles must reflect the user ID's used during the installation.

Level 3 — jcl directory

/rootdir/cloud9/jcl/

Select one or more files with / or action codes.

```
   Type   Perm Changed    (GMT)     Owner         Size     File

 _ Dir    775  10/11/2000 21:21     TCPIP         8192     .
 _ Dir    775  10/11/2000 17:33     TCPIP         8192     ..
 _ File   755  10/11/2000 21:21     TCPIP         2431     CLZJDYN
 _ File   755  10/11/2000 21:22     TCPIP         4791     CLZJIBM
 _ File   755  10/11/2000 21:23     TCPIP         7126     CLZJMIG
```

Level 3 — vaj directory

Select one or more files with / or action codes.

```
   Type  Perm  Changed    (GMT)  Owner            Size  File

 _ Dir    775  03/18/2004 06:42  CLOUD9           8192  .
 _ Dir    755  03/18/2004 06:48  CLOUD9           8192  ..
 _ File   755  03/18/2004 06:42  CLOUD9           3539  b_install.gif
 _ File   755  03/18/2004 06:42  CLOUD9           3527  b_save.gif
 _ File   755  03/18/2004 06:42  CLOUD9           3642  b_saveorrun.gif
```

```
|                    _ File   755  03/18/2004 06:42  CLOUD9           350  bar-fill2.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9           773  bar-left.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          4329  Default.htm
|                    _ File   755  03/18/2004 06:42  CLOUD9          2885  filelist.xml
|                    _ File   755  03/18/2004 06:42  CLOUD9           715  header_fill.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9        212337  image001.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         58197  image002.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          4427  image003.png
|                    _ File   755  03/18/2004 06:42  CLOUD9          9566  image004.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          4566  image005.png
|                    _ File   755  03/18/2004 06:42  CLOUD9          8441  image006.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          9639  image007.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         20142  image008.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          7836  image009.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         19788  image010.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         19788  image010.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         12047  image011.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         31855  image012.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          9047  image013.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         18581  image014.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          7083  image015.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         12792  image016.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          8808  image017.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         29607  image018.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         15209  image019.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         49835  image020.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         10305  image021.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         22883  image022.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          6930  image023.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         15721  image024.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          9277  image025.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         23074  image026.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          7588  image027.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         19062  image028.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9        215730  image029.png
|                    _ File   755  03/18/2004 06:42  CLOUD9        215730  image029.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         62955  image030.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          3434  image031.png
|                    _ File   755  03/18/2004 06:42  CLOUD9          7849  image032.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         35028  image033.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         82223  image034.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9        207788  image035.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         63358  image036.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          3958  image037.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         10028  image038.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          7748  image039.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         19441  image040.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         10090  image041.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         19501  image042.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         11163  image043.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         28832  image044.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          4062  image045.png
|                    _ File   755  03/18/2004 06:42  CLOUD9          8802  image046.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         12055  image047.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         19862  image048.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         19862  image048.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          8322  image049.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         18928  image050.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          4091  image051.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         10171  image052.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9         11676  image053.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         25092  image054.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9        203067  image055.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         60998  image056.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          5088  image057.png
|                    _ File   755  03/18/2004 06:42  CLOUD9         10320  image058.jpg
|                    _ File   755  03/18/2004 06:42  CLOUD9          4622  image059.png
```

```
_ File   755  03/18/2004 06:42  CLOUD9          9720  image060.jpg
_ File   755  03/18/2004 06:42  CLOUD9         96017  image061.png
_ File   755  03/18/2004 06:42  CLOUD9         40490  image062.jpg
_ File   755  03/18/2004 06:42  CLOUD9         18896  image063.png
_ File   755  03/18/2004 06:42  CLOUD9         39736  image064.jpg
_ File   755  03/18/2004 06:42  CLOUD9          5502  image065.png
_ File   755  03/18/2004 06:42  CLOUD9         15756  image066.jpg
_ File   755  03/18/2004 06:42  CLOUD9         49201  image067.png
_ File   755  03/18/2004 06:42  CLOUD9         49201  image067.png
_ File   755  03/18/2004 06:42  CLOUD9         48717  image068.jpg
_ File   755  03/18/2004 06:42  CLOUD9         16222  image069.png
_ File   755  03/18/2004 06:42  CLOUD9         40059  image070.jpg
_ File   755  03/18/2004 06:42  CLOUD9        200184  image071.png
_ File   755  03/18/2004 06:42  CLOUD9         60101  image072.jpg
_ File   755  03/18/2004 06:42  CLOUD9          4124  image073.png
_ File   755  03/18/2004 06:42  CLOUD9         10229  image074.jpg
_ File   755  03/18/2004 06:42  CLOUD9          4292  image075.png
_ File   755  03/18/2004 06:42  CLOUD9          9333  image076.jpg
_ File   755  03/18/2004 06:42  CLOUD9          4481  image077.png
_ File   755  03/18/2004 06:42  CLOUD9         10170  image078.jpg
_ File   755  03/18/2004 06:42  CLOUD9         22132  image079.png
_ File   755  03/18/2004 06:42  CLOUD9         44764  image080.jpg
_ File   755  03/18/2004 06:42  CLOUD9         18807  image081.png
_ File   755  03/18/2004 06:42  CLOUD9         29265  image082.jpg
_ File   755  03/18/2004 06:42  CLOUD9         10261  image083.png
_ File   755  03/18/2004 06:42  CLOUD9         12765  image084.jpg
_ File   755  03/18/2004 06:42  CLOUD9         30602  image085.png
_ File   755  03/18/2004 06:42  CLOUD9         75812  image086.jpg
_ File   755  03/18/2004 06:42  CLOUD9       4100586  install.cab
_ File   755  03/18/2004 06:42  CLOUD9         12401  oci_header.jpg
_ File   755  03/18/2004 06:42  CLOUD9          4976  saving_running.htm
_ File   755  03/18/2004 06:42  CLOUD9       7071579  setup.exe
_ File   755  03/18/2004 06:42  CLOUD9           807  spacer.gif
_ File   755  03/18/2004 06:42  CLOUD9        122224  VAJHELP.htm
```

Note: The profiles must reflect the user ID's used during the installion.

Level 3 — wsad directory

/rootdir/cloud9/wsad/

Select one or more files with / or action codes.

```
  Type   Perm Changed    (GMT)     Owner         Size     File

_ Dir    775  03/18/2004 06:48  CLOUD9          8192  .
_ Dir    755  03/18/2004 06:48  CLOUD9          8192  ..
_ File   755  03/18/2004 06:48  CLOUD9          3539  b_install.gif
_ File   755  03/18/2004 06:48  CLOUD9          3527  b_save.gif
_ File   755  03/18/2004 06:48  CLOUD9          3642  b_saveorrun.gif
_ File   755  03/18/2004 06:48  CLOUD9           350  bar-fill2.jpg
_ File   755  03/18/2004 06:48  CLOUD9           773  bar-left.jpg
_ File   755  03/18/2004 06:48  CLOUD9          1862  filelist.xml
_ File   755  03/18/2004 06:48  CLOUD9           715  header_fill.jpg
_ File   755  03/18/2004 06:48  CLOUD9         53951  image001.png
_ File   755  03/18/2004 06:48  CLOUD9         57509  image002.jpg
_ File   755  03/18/2004 06:48  CLOUD9         11580  image003.png
_ File   755  03/18/2004 06:48  CLOUD9         12117  image004.jpg
_ File   755  03/18/2004 06:48  CLOUD9          5483  image005.png
_ File   755  03/18/2004 06:48  CLOUD9          7626  image006.jpg
_ File   755  03/18/2004 06:48  CLOUD9         58935  image007.png
_ File   755  03/18/2004 06:48  CLOUD9         62662  image008.jpg
_ File   755  03/18/2004 06:48  CLOUD9          7439  image009.png
_ File   755  03/18/2004 06:48  CLOUD9         10032  image010.jpg
_ File   755  03/18/2004 06:48  CLOUD9          7606  image011.png
_ File   755  03/18/2004 06:48  CLOUD9          7606  image011.png
```

```
|                    _ File   755  03/18/2004 06:48  CLOUD9         18874  image012.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          8836  image013.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         18637  image014.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          4850  image015.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         10106  image016.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         21095  image017.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         60802  image018.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          9313  image019.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         21262  image020.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          9333  image021.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         21734  image022.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         58903  image023.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         59313  image024.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         48480  image025.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         80671  image026.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          6779  image027.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         15279  image028.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         58452  image029.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         58467  image030.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         58467  image030.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          8313  image031.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         17745  image032.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         58812  image033.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         62586  image034.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          9080  image035.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         17673  image036.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         57223  image037.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         54919  image038.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          4178  image039.png
|                    _ File   755  03/18/2004 06:48  CLOUD9          9737  image040.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         40700  image041.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         71584  image042.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         94486  image043.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         35879  image044.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         57476  image045.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         41219  image046.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         55492  image047.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         57290  image048.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         58103  image049.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          4096  image050.png
|                    _ File   755  03/18/2004 06:48  CLOUD9          9678  image051.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9         18181  image052.png
|                    _ File   755  03/18/2004 06:48  CLOUD9         24537  image053.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          4329  index.htm
|                    _ File   755  03/18/2004 06:48  CLOUD9         12401  oci_header.jpg
|                    _ File   755  03/18/2004 06:48  CLOUD9          4976  saving_running.htm
|                    _ File   755  03/18/2004 06:48  CLOUD9       8437497  setup.exe
|                    _ File   755  03/18/2004 06:48  CLOUD9           807  spacer.gif
|                    _ File   755  03/18/2004 06:48  CLOUD9         89751  WSADHELP.htm
```

Appendix B. Suite Long Name Registry

The Suite Long Name Registry (SLR) database contains both long name rules and actual data. This is the file where the correlation between the distributed platform object name and the standard z/OS name is maintained. It is referenced in the CIGINI file, in the CLOUD 9 section. The SLR is a standard KSDS VSAM file that needs to be maintained, as all VSAM files need to be maintained.

Note: Before any work is carried out on members in libraries that will have long file name support, ensure that you have defined the NAME RULE for them in the SLR. Defining the rule after members have been created can cause inconsistant results when those members are subsequently modified and saved.

The JCL for CLZSLR

Figure 67 shows sample JCL used to define the SLR long name rules.

```
//**(JOBCARD)
//**
//*********************************************************************
//*                                                                   *
//* THE PURPOSE OF THIS JCL TO RUN THE SLR UTILITY.                   *
//*                                                                   *
//* SEE APPENDIX B, CLOUD 9 FOR SCLM INSTALL GUIDE FOR THE FULL       *
//* SET OF SYNTAX OPTIONS.                                            *
//*********************************************************************
//*                                                                   *
//* REQUIRED JCL MODIFICATION:                                        *
//*   1) INCLUDE A JOBCARD                                            *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//*  STEP 1: ADD DATASET AND TYPE DEFINITIONS TO SLR DATABASE.        *
//*                                                                   *
//*********************************************************************
//STEP1    EXEC PGM=CLZSLR
//STEPLIB DD DSN=CLZ.SCLZLOAD,DISP=SHR
//CIGPUNCH DD SYSOUT=*
//CIGLOG   DD SYSOUT=*
//CIGIN   DD *
 INSERT RULES HERE.
/*
```

Figure 67. Sample SLR Utility JCL

Notes:

1. If the rules you plan to specify in CIGIN, such as ″LIST SHORTNAME WHERE LONG NAME...″, extend past 80 bytes, you must store these rules in a separate data set and then allocate that data set to CIGIN DD. The maximum LRECL for the data set is 256, as the SLR utility does not parse past column 256.
2. If the output expected in CIGPUNCH, such as ’LIST LONG NAME...″, extends past 121 bytes, you must allocate an output data set (with a maximum LRECL of 300) to CIGPUNCH, in order to be able to see the full name.

The utility — CLZSLR

The CLZSLR utility program is used for the following three functions, depending on which syntax is used as input:

1. Add/Delete/List Type definitions for SCLM or Data sets types.
2. Add/Delete/List a Short Name based on a given Long Name.
3. Add/Delete/List a Long Name based on a given Short Name.

The syntax for defining types

This section describes the syntax used for defining types and their attributes. This task is done during initial setup and installation. It is these rules that determine if a transaction is monitored for a distributed object type.

Data set version

```
ADD NAME RULE FOR DATASET 'dataset-name'
case sensitive|case insensitive
.
```

Figure 68. Long name rule syntax for data sets

SCLM version

```
ADD NAME RULE FOR SCLM TYPE 'HostSCM-type'
case sensitive|case insensitive
.
```

Figure 69. Long name rule syntax for SCLM

Keywords for syntax

Table 42. Keywords for long name rule syntax

Keyword	Description	Notes
ADD|DELETE|LIST NAME RULE FOR SCLM TYPE *HostSCM-type*	These keywords and variable are required. The variable is a 1–8 character HostSCM-type that represents a distributed object type.	Required
Case sensitive | case-insensitive	This is an optional keyword that controls the representation of the distributed object name storage. Use of this parameter must reflect the operating system's case sensitivity requirements. For example, UNIX and Linux are case sensitive, whereas Windows files are not.	Default is case insensitive. Ignored for the Delete and List verbs.

Examples of the definition syntax

Data set version

```
ADD NAME RULE FOR DATASET 'A.DEFAULT'.
ADD NAME RULE FOR DATASET 'A.CASE.SENSITIVE' CASE SENSITIVE LRECL 256.
ADD NAME RULE FOR DATASET 'A.CASE.INSENSITIVE' CASE INSENSITIVE.
```

Figure 70. Example of Rule Syntax for Data sets

SCLM version

```
ADD NAME RULE FOR SCLM TYPE XLS .
ADD NAME RULE FOR SCLM TYPE UNIXMAKE CASE SENSITIVE .
ADD NAME RULE FOR SCLM TYPE DOC      CASE INSENSITIVE .
```

Figure 71. Example of Rule Syntax for SCLM

The syntax for adding, deleting, and listing entries in the SLR

This section describes the syntax used for adding, deleting, or listing entries in the SLR. This task is done during processor/translator execution or during any other utility that the user implements.

Short name syntax

```
ADD | DELETE | LIST SHORT NAME WHERE LONGNAME = 'long-name'
  DATASET 'dataset-name' | SCLM TYPE 'sclm-type'.
```

Figure 72. Short Name Syntax

Keywords for short name syntax

Table 43. Keywords for short name syntax

Keyword	Description	Notes
ADD|DELETE|LIST SHORT NAME WHERE LONG NAME = 'long-name'	These keywords and variable are required. The variable is a 1–255 character long name that is translated into a short name for the ADD.	The long name entry must exist for the DELETE and either case can be true for the LIST function. You cannot use wild cards in the name.
DATASET 'dataset-name' | SCLM TYPE 'sclm-type'	This keyword further defines the attributes of the names.	Required.

Note: The 'long-name' value must be specified on a single line, as line wrapping is not supported. If necessary, these values can be stored in a data set that has an LRECL of up to 256 bytes. The data set must then be allocated to CIGIN DD.

Examples of short name syntax

```
delete short name where longname = 'Fiscal Year End 2000 Spread Sheets.xls'
  SCLM type xls.
```

Figure 73. Example of Short Name Syntax for SCLM

Figure 74 is an example of the output generated by a LIST Short Name Request. The short name is displayed first, with an asterisk in column 1.

```
* HEL00001
LIST SHORTNAME WHERE LONGNAME =
HELLO STEVE.
```

Figure 74. Example of LIST Short Name Output

Long name syntax

```
LIST LONG NAME WHERE SHORTNAME = 'short-name'.
```

Figure 75. Long Name Syntax

Keywords for long name syntax

Table 44. Keywords for long name syntax

Keyword	Description	Notes
LIST LONG NAME WHERE SHORT NAME = 'short-name'	These keywords and variable are required. The variable is a 1–8 character short name.	You cannot use wild cards in the name.

Examples of long name syntax

```
list longname where shortname = 'HEL00001'.
```

Figure 76. Example of long name syntax

Figure 77 is an example of the output generated by a LIST long name request. The long name is displayed first, with an asterisk in column 1.

```
* HELLO STEVE
LIST LONGNAME WHERE SHORTNAME = HEL00001 .
```

Figure 77. Example of LIST Long Name Output

SLR in SCLM translators

The z/OS SCLM tool is not aware that the short name stored in its repository is actually a long name somewhere else. From a programming object perspective, the short name is a fully-qualified member and normal object being tracked and promoted. As the short name is moved up the inventory maps, the reference to the long name still exists in the SLR. When the user lists against these from the Cloud 9 Browser interface, the long names are displayed.

Appendix C. CA-Endevor Bridge customization

There is an CA-Endevor Bridge exit point in Cloud 9 that might need to be the modified.

CLZREXIT - C1UXSITE support

CLZREXIT is the rexx exec for C1UXSITE, multiple C1DEFLTS switching. Review this exit program for customization. The sample exit program can be found in the cgi-bin directory of your Cloud 9 rootdir.

```
/* rexx ---------------------------------------------------------------
   - Program: CLZREXIT                                                -
   - Purpose: This is the exit driver program.                        -
   -                                                                  -
   - Exit 1: Is ENUXSITE to be called?                                -
   -    return ''                   do not call ENUXSITE              -
   -    return 'YES'                call ENUXSITE                     -
   -    return 'YES,DDNAME,DSNAME'  call ENUXSITE and allocate        -
   -                                ddname/dsname before call         -
   -                                                                  -
   - Exit 2: Is CIGINI loader to be called?                           -
   -    return ''                   do not call CIGINI loader         -
   -    return 'PGM'                call specified program            -
   -    return 'PGM,DDNAME,DSNAME'  call specified program and        -
   -                                pass ddname/dsname to             -
   -                                specified program.                -
   -                                                                  -
   ------------------------------------------------------------------- */
    parse arg xitno
    select
        when (xitno == '1') then buffer = Exit01()
        when (xitno == '2') then buffer = Exit02()
    end

return buffer
```

Figure 78. CLZREXIT (Part 1 of 2)

```
/* ------------------------------------------------------------- */
/*   *** C1DEFLTS table switching ***                            */
/*   ENUXSITE gets called when requesting a list of environments */
/*   or calling Endevor to perform an action.                    */
/* ------------------------------------------------------------- */
Exit01:
    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    /*  Example: Do not call ENUXSITE.                           */
    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    buf = ''

    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    /*  Example: Invoke ENUXSITE, but do not allocate a ddname.  */
    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    /*  buf = 'YES'                                              */

    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    /*  Example: If the user is P390Z then invoke ENUXSITE and   */
    /*           allocate the specified ddname/dataset name prior*/
    /*           to invoking ENUXSITE.                           */
    /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    /*  if (userid() == 'P390Z') then                            */
    /*      buf = 'YES,CIGDD01,CIGT.STEVE.LOADLIBX'              */

return buf
/* ------------------------------------------------------------- */
/*   *** CIGINI switching ***                                    */
/*   This logic is used if the FastLIST database is used to get a*/
/*   list of systems, subsystems, types, or processor groups. It is */
/*   also used when the FastLISt database is used to get a list of */
/*   elements.                                                   */
/* ------------------------------------------------------------- */
Exit02:
    /*  Example: Do not switch CIGINI files.                     */
        buf = ''

    /*  Example of calling exit program called CIGXSAMP          */
    /*  buf = 'CIGXSAMP'                                         */

    /*  Example of calling exit program called CIGXSAMP          */
    /*  and have the following dd statement allocated.           */
    /*  //CIGDD01  DD DSN=CIGT.STEVE.LOADLIB,DISP=SHR           */
    /*  buf = 'CIGXSAMP,CIGDD01,CIGT.STEVE.LOADLIB'              */

return buf
```

Figure 78. CLZREXIT (Part 2 of 2)

A compliment to switching C1DEFLTS is switching CIGINI files.

Index

Special characters

A

B

C

D

E

F

H

I

J

L

M

O

P

R

S

T

U

V

W

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504–1785, USA.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries in writing to

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, 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 OR NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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 the IBM Corporation, Department TL3B, 3039 Cornwallis Road, Research Triangle Park, North Carolina, 27709–2195, USA. 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.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

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

- IBM
- SCLM
- z/OS
- WebSphere

Breeze, Cloud 9, and SOLEI are trademarks of Chicago Interface Group, Incorporated.

Internet Explorer and Windows are trademarks of Microsoft Corporation.

Netscape Navigator is a trademark of Netscape Communications Corporation.

CA-Endevor is a trademark of Computer Associates, Inc.

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

Program Number: 5655-G93

Printed in USA

SC31-8845-05