
Chicago Interface Group, Inc.

Cloud 9 for SCLM Administration Guide



Version 2.1

Chicago Interface Group, Inc.
858 West Armitage Avenue #286
Chicago, IL 60614 USA

Phone: (773) 524-0998
Fax: (815) 550-6088
EMAIL: support@cigi.net
WEB: www.cigi.net

Cloud 9 is a trademark of Chicago Interface Group, Inc.
SCLM is a registered trademark of IBM.

All rights reserved. © Copyright by Chicago Interface Group, Inc., 2007

Cloud 9 Release 2.1.

Documentation Version August 16, 2005

CONTENTS

Chapter 1: Getting Started.....	6
Introduction.....	6
Who should use this manual?	6
How to use this manual?.....	6
Seeing the Big Picture.....	7
Required Tasks for all Cross Platform Applications	7
Optional Tasks for a Cross Platform Application:.....	8
Chapter 2: Standard Cross Platform Setup	9
Step 1: Define File Types to SCLM	9
SCLM Base File Requirements:	9
SCLM Version File Requirements:	9
Step 2: Define the Type to SLR.....	10
Step 3: Add Type Extension to the z/OS HTTP Rules File (httpd.conf).....	12
Step 4: Update YOUR Browser's File Type Settings.....	14
Internet Explorer MIME Setup:.....	14
Netscape MIME Setup:.....	16
Chapter 3: The SLR Utility	17
Chapter Scope	17
Long Name Support.....	17
The Utility – C9LSLR	18
The JCL for C9LSLR.....	18
The Syntax for Defining Types to the SLR	19
Example of SLR Definition Syntax:.....	20
The Syntax for Adding, Deleting, and Listing Entries in the SLR.....	20
Example of SLR Short Name Syntax:	21
List Short Name CIGPUNCH Example:	21
Example of SLR Long Name Syntax:	21
List Long Name CIGPUNCH Example:.....	21
SLR in SCLM Translators	22
Using C9LSLR with SCLM.....	22
Chapter 4: S-FTP Remote Build and Deployment	23
Chapter Scope	23

A Step-by-Step Approach.....	24
Product Components that require modification	24
JCL Members (located in JCL) Modified During Implementation:	24
Review Software Requirements and Assumptions	25
Step 1. Review and Set S-FTP Inventory Values	26
FTP Target Platform Worksheet.....	28
Step 2. Review SCLM and Cloud 9 Type definitions	29
Type Review Matrix	29
Determining Types Exist?.....	29
Determining Cloud 9 Definitions?.....	29
Determining LRECL and File Attributes?.....	29
CHECKPOINT #2	30
Step 3. Review and modify translator and REXX script	31
Review and modify the CIGRFTP1 REXX Script	33
Step 4: Update Your Project Definition.....	37
Example of Project Definition Update.....	37
CHECKPOINT #3	38
Step 5. Test the S-FTP Translator.....	39

III. S-JDK SCLM Java Development Kit.....43

Chapter Scope	43
Warning – Read First	43
Other Documents For Reference	44
Verify JAVA/USS Environment.....	44
A Step-by-Step Approach.....	45
Before You Begin: Review Software and Assumptions.....	46
Assumptions:.....	46
Step 1. Determine SCLM and USS Inventory Values.....	47
SCLM Inventory Value Worksheet	47
USS Directory Value Worksheet	47
Step 2. Review and Document SCLM and Cloud 9 Type definitions	48
Type Review Matrix	48
Determining Types Exist?.....	49
Determining Cloud 9 Definitions?.....	49
CHECKPOINT #4	50
Step 3. Review and modify all Translators and Control Files	51
Review and modify CIGTJAVA translator	51
Review and modify CIGTJAR Translator	52
Review and modify CIGTJTXT translator	54
Review and modify CIGTJBIN translator	55
Review and modify CIGTULOC – Common SCLM to USS Life Cycle Map	56
Modify CIGTCPTH – Common %CLASSPATH% Substitution:	56
Modify CIGTJAVC – JAVA Compile Shell:.....	57
Modify CIGTUMAP Java Output Mapping Rules:.....	57
Modify CIGTJCOMP Jar Compile Shell:	59
Modify CIGTJMAP - Jar Output Mapping Rules:	59

CHECKPOINT #5	60
Step 4: Update the Project Definition	61
Project Definition Updates.....	61
Allocate SCLM Type Files – CIGTALIB	63
Allocate Project VSAM Files - CIGTAVSM	66
Step 5: Define S-JDK types to Cloud 9 SLR.....	69
Modify and Submit CIGC9J06.....	69
Step 6: Run CIGJAUNX to build S-JDK USS Directories	70
Modify and Submit CIGJAUNX	70
Step 7: Review CIGJCIG Unix Shell – Delete Processing.....	72
Understanding SCLM Delete Processing	72
CHECKPOINT #6	75
Step 8. Test the S-JDK Translators.....	76
Step 9: Invoking the Compiled Java.....	78
Result of Clock2.java Compile:.....	78
CHECKPOINT #7	79
Chapter 6: Cloud 9 Exits	80
CLZREX00 – Cloud 9 Temporary Dataset Prefix Setting	80
Appendix A - Type Definition Worksheet	81
Appendix B - Application Life Cycle Worksheet.....	82
Application Life Cycle Requirements	82
Appendix C - FTP Deployment Worksheet.....	83

Chapter 1: Getting Started

Introduction

Who should use this manual?

The audience for this manual is technicians and administrators responsible for the configuration of Cloud 9. It assumes basic knowledge of HTTP server concepts, knowledge of SCLM and a general knowledge of browser setup features. In short, there may not be one person at a customer site that knows all of these things. So this manual is meant as a starting point or perhaps a joining point for these various technical issues.

How to use this manual?

This manual outlines some common aspects of using Cloud 9 to manage cross platform objects. There is an introduction section, information on long names, specifics on setting up cross platform objects, FTP-based deployment and remote builds, and the Cloud 9 S-JDK prototype translator. Use this manual to get started in the planning and configuration process of Cloud 9 for SCLM.

Seeing the Big Picture

Before taking steps to implement a complex, cross platform application using Cloud 9, it would be helpful to take a step backwards and break up the process into smaller buckets. The process can be divided into three categories: 1) The issues that will be the same and required for all applications; 2) The issues that are unique per application; and 3) The unknowns that will have to be sorted out when we get there.

As an exercise moving forward you should look at each application from this perspective. What is the fixed, known, absolutely required work for each application, then what additional scripts and functionality can be customized for the application, and thirdly, what additional things could the end user ask for potentially.

Required Tasks for all Cross Platform Applications

Starting with the standard issues that will be required for each application:

1. Like any SCM implementation, if this is a new application to change management you will need to define the types and attributes of each type. For instance, Visual Basic files such as .FMX or .FMT will need an Endeavor or SCLM type associated with them on the host. See Appendix A for an example of a Type Definition Worksheet
2. Once the types are identified for the application, per type, what kind of control is needed for the object? Version control only, deployment of object to servers, a remote build requirement? Again see Appendix B for an example of an FTP worksheet and Appendix C for the Application Life Cycle Requirements Worksheet.
3. Build the SCLM types into the Projdefs and allocate the files. It is recommended that you use a batch job for this purpose.
4. Define the types to the SLR. See Chapter 3 on more about the SLR long name utility.
5. Define the types to your workstations and HTTP server. Essentially, your workstation needs to know how to handle a file extension when it is delivered from a browser. Much like your email interface reacts to an attachment, the Cloud 9 browser interface needs to know the application associated with the file extension. The HTTP server needs to know this as well. Most common file extensions are already defined to your workstation and to the HTTP server. Please see chapter 2, Managing Cross System Applications for more on this issue.

Optional Tasks for a Cross Platform Application:

Having worked through the Type and Process Matrixes, there should be an idea of your success criteria. What are you hoping to accomplish with this application? Here are some of the questions to ask:

1. Do you need a translator to deploy an executable or html file outside of SCLM? Can you accomplish what you need by implementing and customizing the FTP translators included with Cloud 9? See Chapter 4 for more information on the S-FTP translators.
2. Will you be compiling Java on the host using USS? Do you want absolute correlation between source and executable, much like standard SCLM map relationships? Please review Chapter 5 for more information on the Java/USS translators.
3. Do you want to do remote builds? What does this really mean? Typically the problem is that production control minimally wants to lock down production and would like to, if possible to force the creation of the executable from the source saved in the host repository. This can be accomplished after researching the source type and how it is compiled today. This area will always need customization and possibly some research. Cloud 9 provides an S-FTP translator that shows a simple C++ make file being sent to a remote box for execution. Most remote builds will require REXX script customization. Also note that not all IDE applications have a batch or command line interface for requesting builds. You will need to understand the nature of the IDE to determine if it is eligible for remote build processing.

Chapter 2: Standard Cross Platform Setup

Regardless of which cross platform applications you chose to support with Cloud 9, there are four basic steps that must be performed before the application can be supported. Two of the steps are standard z/OS batch jobs, one is a potential update to the HTTP server files, and finally, the last is an update to your actual workstation. Review this section for the simple steps to setting up a cross platform application.

Step 1: Define File Types to SCLM

1. Determine the type name. It is recommended that you name the type the same as the file extension. For instance, *.DOC* types should be defined as *DOC*, *.JAVA* types should be defined as *JAVA*, etc. See Appendix A for a sample type definition work sheet.
2. Define the SCLM Base and Version files per Cloud 9 requirements.

SCLM Base File Requirements:

```
LRECL=256  
BLKSIZE=27998  
RECFM=VB
```

SCLM Version File Requirements:

```
LRECL=350  
BLKSIZE=27998  
RECFM=VB
```

3. Define the type to SCLM using the PROJDEFS, just like any other type. The following is an example of the required type attributes for cross platform types. Due to the undefined and variable lengths of the most non-z/OS files, the lengths are generally longer than host files. Also, these attributes are designed to work with the base and version as defined in this section.

Step 2: Define the Type to SLR

Run the SLR update utility to define the type to the SLR. The figure below shows an example of how to list the current types and how to define a new type. For more in-depth syntax and usage of the SLR utility, please see Chapter 3, of this manual.

Note there is no harm in re-defining and already existing type.

```
/***(JOB CARD)
/**
*****
/**
/** CIGSOJ02 - THE PURPOSE OF THIS JCL IS TO RUN THE SLR UTILITY.
/**          STEP 1 WILL PRINT THE CIGINI.
/**          STEP 2 HAS EXAMPLES OF LISTING, ADDING AND DELETING
/**          TYPE DEFINITIONS.
/** NOTE:    - SEE THE CLOUD 9 V2 PLANNING AND ADMINISTRATION GUIDE
/**          FOR MORE INFORMATION ON LONGNAME SETUP AND USAGE.
/** NOTE:    - THE SYNTAX PROVIDED IS FOR AN EXAMPLE ONLY.
/**          IT IS RECOMMENDED THAT STEP3 SYNTAX BE TAILORED TO
/**          ACTUAL LOCAL VALUES AND SCLM TOOL.
*****
/**
/** REQUIRED JCL MODIFICATION:
/** 1) INCLUDE A JOB CARD
/** 2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.
/**    - FLHQ1 AND FLHQ2
*****
/**
/** STEP 1: PRINT THE CIGINI DEFINITIONS.
/**
*****
//STEP1      EXEC PGM=PRINTINI
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGPRINT DD SYSOUT=*
*****
/**
/** STEP 2: LIST THE CURRENT CONTENTS OF THE SLR DATABASE
/**
*****
//STEP2      EXEC PGM=C9LSLR
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGPUNCH DD SYSOUT=*
//CIGLOG DD SYSOUT=*
//CIGIN DD *
LIST NAME RULES.
ADD NAME RULE FOR DATASET 'FLHQ1.FLHQ2.PDS1' CASE SENSITIVE.
ADD NAME RULE FOR DATASET 'FLHQ1.FLHQ2.PDS2' CASE INSENSITIVE.
ADD NAME RULE FOR SCLM TYPE HTML .
ADD NAME RULE FOR SCLM TYPE UNIXMAKE CASE SENSITIVE .
ADD NAME RULE FOR SCLM TYPE DOC CASE INSENSITIVE .
ADD NAME RULE FOR ENDEVOR TYPE HTML .
ADD NAME RULE FOR ENDEVOR TYPE UNIXMAKE CASE SENSITIVE .
ADD NAME RULE FOR ENDEVOR TYPE DOC CASE INSENSITIVE .
```

```
LIST NAME RULES.  
/*
```

1. CIGSOJ02

Step 3: Add Type Extension to the z/OS HTTP Rules File (httpd.conf)

Check the httpd.conf file to see if the file extension your adding is already there. The following is the ADDTYPE table delivered with the Cloud 9 version of the httpd.conf, which can be found in the “rootdir” of the Cloud 9 USS directories.

```
#-----  
#  
#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
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 Enhanced 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

```

2. Example of ADDTYPE Entries

If the file type your adding is not there, then add it using the following format:

AddType / Extension / Mime type / Translation Technique MIME Definition Format

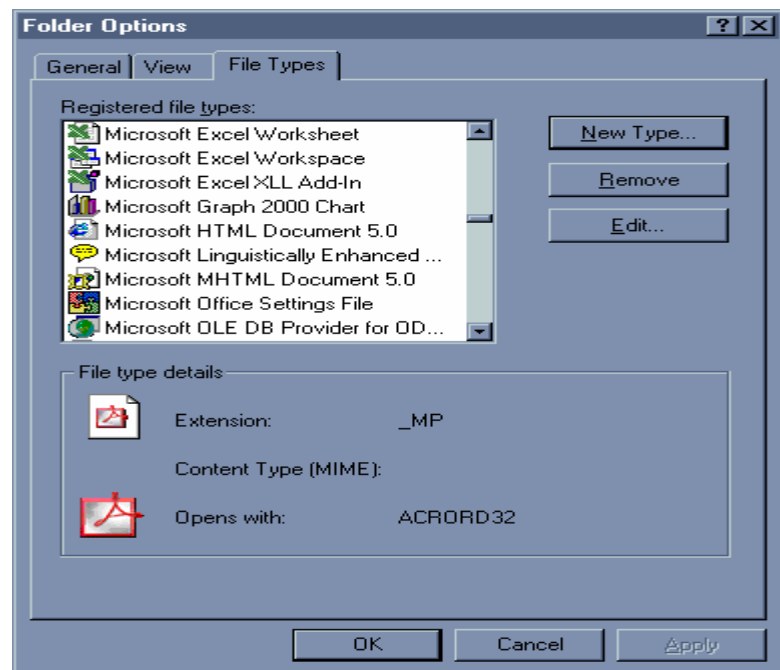
For example if adding an *MS-Excel* file type, the following format would be used:

```
AddType .xls application / msexcel binary
```

Step 4: Update YOUR Browser's File Type Settings

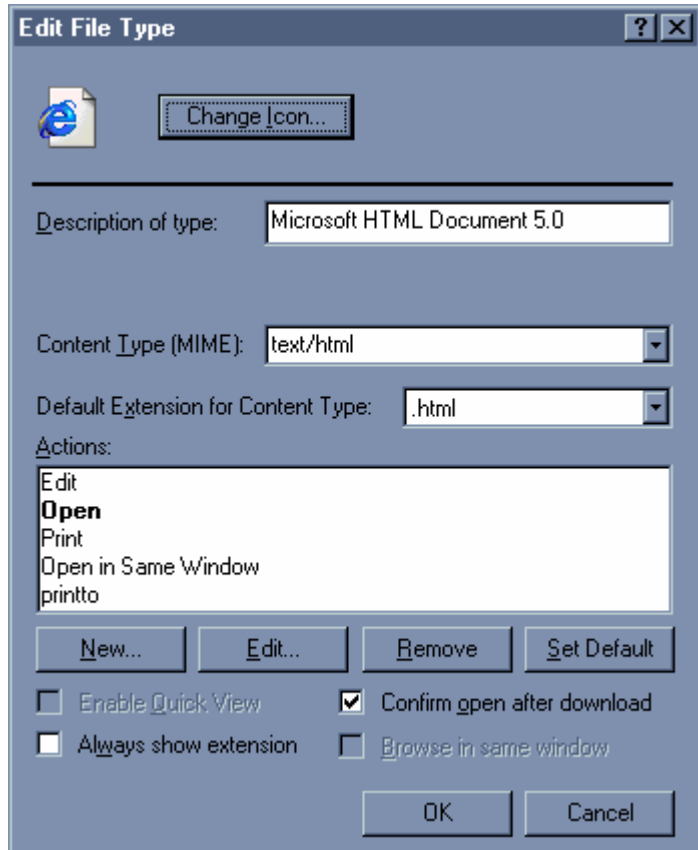
Internet Explorer MIME Setup:

On Windows, go to Start / Settings / Folder Options / File Types.



3. Folder Options

Check the list of file types for the file type you will be downloading. If the file type you're looking for is there, then the application currently set to open the file will be displayed. If the file type is there, but set to the wrong application, then select Edit.



4. Edit File Type

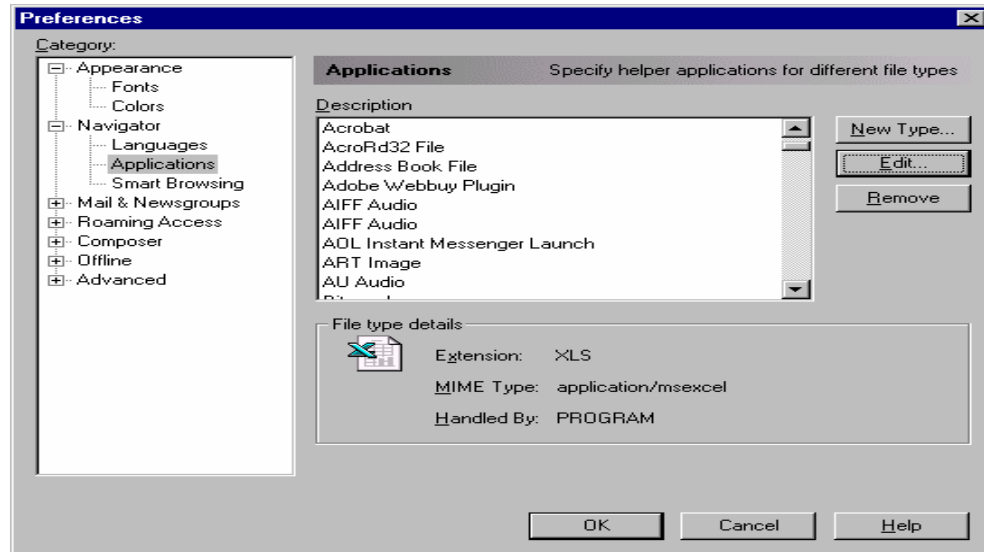
On the Edit screen you can specify what application is chosen to open the file. Also, the “Confirm open after download” option gives you the choice of whether or not a prompt will occur after a download.

If the file type your looking for is not in the file list then click the “New Type” button from the Folder Options screen

This screen allows you to add a file type to the file list and choose a default application to open the file with. Once the file type has been edited or added, the **httpd.conf** file should be checked to make sure that all the **ADDTYPE** definitions match.

Netscape MIME Setup:

In Netscape, go to Edit / Preferences / Navigator / Application.



5. Netscape Folder Options

Check the list of file types for the file type you will be downloading. If the file type you are looking for is there, then the application currently set to open the file will be displayed.

Ensure that the correct application is set up to open your file. If it is not set to the right application then select “Edit”.

If the file type you are looking for is not in the list of file types, then select “New Type”.

This screen allows you to add a file type to the file list and choose a default application to open the file with.

Once the file type has been edited or added, the httpd.conf file should be checked to make sure that all the ADDTYPE definitions match.

In Netscape, any file without an extension is given a default extension of .TXT. To change this default extension, you must change the “Handled by” option for the file types with the description, “plain text”.

Chapter 3: The SLR Utility

Chapter Scope

- How long name support works
- The SLR database

Long Name Support

Cloud 9 is a solution for managing cross platform objects using SCLM. One of the anchor pieces of Cloud 9 is long name support, required when moving, viewing and referencing objects from one platform to another.

The premise behind long name support is that each customer will selectively decide which SCLM types are monitored and managed by Cloud 9. Your z/OS system will have both standard host based and cross platform objects managed side by side, there needs to be a method to signal Cloud 9 that short-to-long name file renaming occur. This task is accomplished through the use of a VSAM file called the Short-to-Long Name Registry or SLR.. CLOUD 9 references the SLR when source is added or updated; building the appropriate SCLM commands.

The SLR database contains both long name rules and actual data. This is the file where the correlation between the distributed platform objects' name and the standard z/OS eight-character name is maintained. It is referenced in the CIGINI file, within the CLOUD 9 Section. The SLR is a standard KSDS VSAM file that will need to be maintained as all VSAM files are maintained.

The Utility – C9LSLR

The utility program C9LSLR is used for the following three functions, depending on which syntax is used as input:

1. Add/Delete/List Type definitions for SCLM.
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 JCL for C9LSLR

The following is the JCL used to define, manage and report the SLR long name rules and data to the CIG Suite. It can be found in the Cloud 9 JCLLIB offloaded from the installation tape.

```
/** (JOB CARD)
/**
/**-----*
/*******
/**
/** CIGSOJ02 - THE PURPOSE OF THIS JCL IS ADD LONG NAME RULES TO THE *
/**          SLR DATABASE FOR CIG SUITE                               *
/**          *                                                       *
/** NOTE:    - SEE THE CIG SUITE SOLUTION GUIDE FOR INFORMATION    *
/**          ON LONG NAME RULE SYNTAX AND USAGE.                   *
/*******
/**
/** REQUIRED JCL MODIFICATION:                                       *
/** 1) INCLUDE A JOB CARD                                           *
/** 2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.     *
/**    - FLHQ1 AND FLHQ2                                           *
/**    - VOLUMES (DVOLSER)                                         *
/** 3) INSERT LONG NAME RULES SYNTAX AS PER REQUIREMENTS.         *
/**          *                                                       *
/**-----*
/**
/** STEP 1: ADD SLR CONTROL STATEMENTS TO THE SLR DATABASE
/**
/**STEP1 EXEC PGM=C9LSLR ADD CONTROL RECORDS TO SLR
/**STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
/**CIGIN DD *
/**      ADD NAME RULE FOR . . . PUT ADD STATEMENTS HERE
/**CIGPUNCH DD SYSOUT=*
/**CIGLOG DD SYSOUT=*
```

6. CIG SUITE Utility C9LSLR

The Syntax for Defining Types to the SLR

The following is the rules syntax used for defining types and their attributes to the SLR. This task would be done during initial setup and installation. It is these rules that determine if the CIG Suite will monitor the transaction for the distributed object type. These rules are used in the CIGIN DD statement.

```
ADD NAME RULE FOR SCLM TYPE 'HostSCM-type'
case sensitive|case insensitive
LRECL lrecl-length.
```

7. SLR Long Name Rule Syntax for SCLM

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 sensitivel Case insensitive	This is an optional keyword that controls the representation of the distributed object name storage. Usage of this parm should reflect the platform case sensitivity requirements. For instance, Unix and Linux is case sensitive, where as Windows files are not.	Default is case insensitive. Ignored for the Delete and List verbs.
LRECL <i>lrecl-length</i> .	This is an optional keyword that controls the DCB LRECL length of the host staging dataset. The <i>lrecl-length</i> should reflect the length of the type in the host SCM tool.	Default is 80 bytes. Ignored for the Delete and List verbs.

8. SLR Long Name Rules Parameter Description

Example of SLR Definition Syntax:

The following is an example of SCLM based rule definitions.

```
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
.
```

9. SLR Definition Rule Syntax Example

The Syntax for Adding, Deleting, and Listing Entries in the SLR

The following is the syntax used for creating, deleting or listing entries in the SLR. This task would be done during processor / translator execution or during any other utility that the customer implements.

Short Name Syntax:

```
ADD|DELETE|LIST SHORT NAME WHERE LONGNAME = 'long-name'
SCLM TYPE 'sclm-type'
```

10. SLR Short Name Entry 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 will be 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. Wildcarding is not allowed.
SCLM TYPE 'Sclm-type'	This keyword further defines the attributes of the names.	One of these keywords are required for the ADD and DELETE verbs but are optional for the LIST verb.

11. SLR Short Name Rules Parameter Description

Example of SLR Short Name Syntax:

The following is an example of SCLM based short name request.

```
LIST SHORT NAME WHERE LONGNAME = 'HelloWorld.java' .
```

12. SLR Short Name Syntax Example

List Short Name CIGPUNCH Example:

The following figure is an example of the output generated into the CIGPUNCH DD by the LIST Short Name Request. The short name appears first with an asterisk in column 1.

```
* HEL00001  
LIST SHORTNAME WHERE LONGNAME = 'HelloWorld.java' .
```

13. List Long Name Output

Long Name Syntax:

```
LIST LONG NAME WHERE SHORTNAME = 'short-name' .
```

14. SLR Long Name Entry 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.	Wildcarding is not allowed.

15. SLR Long Name Rules Parameter Description

Example of SLR Long Name Syntax:

The following is an example of SCLM based longname request.

```
LIST LONGNAME WHERE SHORTNAME = 'HEL00001' .
```

16. SLR Long Name Syntax Example

List Long Name CIGPUNCH Example:

The following figure is an example of the output generated by the LIST LONG NAME request. The long name appears first with an asterisk in column 1.

```
* HelloWorld.java  
LIST LONGNAME WHERE SHORTNAME = HEL00001 .
```

17. List Long Name Output

SLR in SCLM Translators

The SCLM 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. For customers who will be using Cloud 9 solution purely as a cold storage mechanism, meaning that no additional processing will be done against the element or member while on the host, then no additional processor or translator work is required. 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 will appear.

Using C9LSLR with SCLM

As shown in the previous section of this chapter, the C9LSLR utility is a standard utility that can be used in an SCLM translator. This utility would primarily be used as a lookup service for generating FTP statements or component lists. The information returned in the CIGPUNCH DD from the lookup request needs to be parsed and processed as per requirements.

Chapter 4: S-FTP Remote Build and Deployment

Chapter Scope

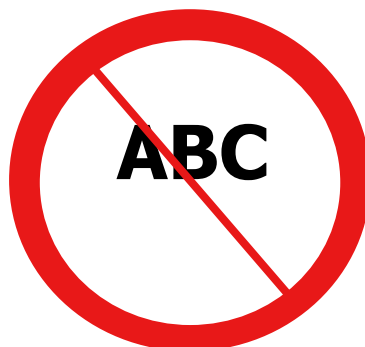
This chapter contains the instructions to customize and implement Cloud 9 S-FTP, the FTP based Deployment and Remote Build Scripts. The steps in this chapter are organized into five sections:

- Reviewing software requirements and assumptions.
- Review the S-FTP SLR and SCLM defaults and setup
- Customizing translator and REXX Script
- Defining the S-FTP translator to your SCLM Project Definition.
- Test the S-FTP Translator

These steps should be followed in the order that they are presented. Once you have successfully completed all of the steps and executed the test procedure in Step 5, you will be ready to use the S-FTP translators for remote builds and deploys.

CASE SENSITIVITY ALERT!

IMPORTANT: During installation, you will modify JCL members and REXX Scripts. Some of these files contain case-sensitive values. It is imperative that *prior* to modifying the members, you issue the CAPS OFF command to ensure that automatic upper casing of the members does not occur. For your convenience, the following icon will be placed in each step where case-sensitive values are an issue.



A Step-by-Step Approach

BEFORE YOU BEGIN...	
◆	Review system, software, and hardware considerations.
Review Default SCLM Inventory Locations and USS locations	
1.	Review default S-FTP values and determine actual inventory and FTP targets to be used.
2.	Review SLR and SCLM definitions for FTP supported TYPES.
3.	Review and modify the FTP translator and REXX Script.
Perform SCLM and Cloud 9 Definitions	
4.	Include the CIG@FTP1 translator in your SCLM Project Definition.
Test Translator using Sample HTML file	
5.	Add an HTML file and perform a build on the file.

18. Enabling the S-FTP

Product Components that require modification

The following JCL members are modified during the implementation process. The names are provided here as an overview of CIG naming standards and component functionality.

JCL Members (located in JCL) Modified During Implementation:

CIG@FTP1	Cloud 9 FTP Translator
CIGRFTP1	REXX Script

Review Software Requirements and Assumptions

In this step you will review the system and software requirements for enabling the S-FTP.

System Requirements:

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

z/OS Operating System	Version 1.1 or later
z/OS FTP Server	Standard z/OS
Target FTP Server	Specific to platform

Assumptions:

This solution guide does not cover the definitions of types to SCLM. This is covered in many SCLM manuals and is outside the scope of this manual.

This solution guide does not cover configuring FTP on the z/OS platform or any of the target FTP platforms.

This solution guide assumes requires that you identify which types are to be deployed to remote locations. It is assumed that the types have been defined in the project definition and the SLR for cross platform types. Step 4 conducts a review of both of these concepts.

Rexx Knowledge Required:

The CIGRFTP1 Rexx Script requires modifications. This manual provides guidelines; however, the modifications are best performed by someone who has Rexx experience. This is not a canned, black box modification procedure.

FTP Server Requirements:

This manual does not cover installing and configuring the FTP servers on the z/OS or any of the targets. This task is probably already complete. If not, the network system's programmers will have to be contacted to perform this work.

Step 1. 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. Prior to making modifications to these members, please review the worksheets below and fill in the site specific Inventory Values.

It is important to review and record all possible targets for builds and remote deployment.

SCLM Inventory Name	Default Value	Your Values
SCLM Group	DEV, TEST are used as default group names in CIGRFTP1.	
SCLM MAKE Type Referenced	MAKE is referenced as the default 'make' type in CIGRFTP1.	
SCLM Binary Types Referenced	GIF, JPG are referenced as examples of types that have binary FTP attributes in CIGRFTP1.	
SCLM Language	FTP1 – set in CIG@FTP1.	
User	Use rid is used in CIGRFTP1 in several places.	
Password	Pass is used in CIGRFTP1 in several places.	
Dir	Used in all four of the Set Target Examples in CIGRFTP1. For USS: '/u/userdir' For ISP: '/isp/userdir' For C drive: 'c:\userdir' For A drive: 'a:\userdir'	
Ip-address	The ip-address is set up as a dummy value of 999.999.999.99 in all four of the Set Target	

	Examples in CIGRFTP1.	
Port	The Port is set to a default of 21 in all four of the Set Target Examples CIGRFTP1.	
Userlog	Userid.ftplog is used at the end of CIGRFTP1. It is the default dataset name for the FTP log file.	

19. S-FTP Value Worksheet

Step 2. Review SCLM and Cloud 9 Type definitions

Prior to moving on to the Rexx and Translator modification, it is advised that all types selected for deployment be reviewed. First fill in each unique type in the Type Review Matrix below. Then for each type, review the definition in SCLM and Cloud 9 SLR for existence, consistency and correctness.

Also, it is recommended that the FTP target directories also be defined. Use the previous “FTP Target Platform Worksheet” to document that the target directories are defined. You may also need to meet with your Network Administrator to review security for the target platforms.

Type Review Matrix

Type	Language is FTP1	Defined to SCLM	Defined to Cloud 9 SLR	Case Sensitive	Binary (lrecl = 256)

21. Type Review Matrix

Determining Types Exist?

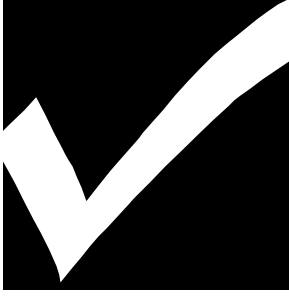
To determine if a type exists, go to Cloud 9, list SCLM members. For the Group, list the types and verify that the types are there. If they do not exist, then you must define the types and datasets to SCLM. The language should not be there yet.

Determining Cloud 9 Definitions?

To determine if the SCLM type is defined to the Cloud 9 SLR, run the IVP JCL member CIGC9J06. Review the output from the job, verifying 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 TSO Option 3.2 and display the dataset information for one of the type datasets. If binary the LRECL will be 256 and the RECFM = VB.



CHECKPOINT #2

At this point the following tasks should have been completed.

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

22. Checkpoint 2

Step 3. Review and modify translator and REXX script

The following sample translator can be found in the JCLLIB library offloaded during the product install:

1. Using ISPF EDIT, access member CIG@FTP1 in the JCLLIB library. There is not a lot of modification for this. The translator uses default CIG naming standards for all product files.
2. Ensure that the flqh1.flhq2.JCLLIB dataset is in the SYSLIB concatenation of Project Definition JCL. Either include the dataset or copy this member to a library in the SYSLIB concatenation.

```
*****
* NAME:      CIG@FTP1
* PURPOSE:   INVOKE CLOUD 9 FTP REXX SCRIPT TO DEPLOY AND/OR BUILD
*            REMOTE OBJECTS. ( SCRIPT NAME IS CIGRFTP1)
*****
* NOTE:      THIS PROTOTYPE TRANSLATOR REQUIRES CUSTOMIZATION
*            BY THE CUSTOMER OR INSTALLER.
*            THE ACTUAL DEPLOYMENT OR BUILD REQUEST IS PERFORMED
*            BY THE REXX EXEC CIGRFTP1. THIS REXX EXEC NEEDS TO BE
*            CUSTOMIZED TO MEET THE INVENTORY NAMES AND TARGET
*            LOCATIONS REQUIRED BY THE CUSTOMER.
*****
* CHANGE ACTIVITY:
*
*
*****
                FLMLANGL      LANG=FTP1,VERSION=TEXTV1.0
*****
* PARSER TRANSLATOR
*
*****
                FLMTRNSL      CALLNAM='SCLM TEXT PARSE',
                                FUNCTN=PARSE,
                                COMPILE=FLMLPGEN,
                                PORDER=1,
                                OPTIONS=(SOURCEDD=SOURCE,
                                STATINFO=@@FLMSTP,
                                LISTINFO=@@FLMLIS,
                                LISTSIZE=@@FLMSIZ,
                                LANG=T)
*            (* SOURCE          *)
                FLMALLOC      IOTYPE=A,DDNAME=SOURCE
                FLMCPYLB      @@FLMDSN(@@FLMMBR)
*****
* BUILD TRANSLATOR
*
* STEP 1A: GET THE LONGNAME FROM THE SLR
*****
                FLMTRNSL      FUNCTN=BUILD,CALLNAM='STEP1A: CIGFPARM',
                                COMPILE=CIGFPARM,DSNAME=FLHQ1.FLHQ2.LOADLIB,
*****
```

```

        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: CIGFCOPY',          X
        COMPILE=CIGFCOPY,DSNAME=FLHQ1.FLHQ2.LOADLIB,             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: C9LSLR',  X
        COMPILE=C9LSLR,DSNAME=FLHQ1.FLHQ2.LOADLIB
        FLMALLOC DDNAME=CIGLOG,IOTYPE=W
        FLMALLOC DDNAME=CIGIN,IOTYPE=U
        FLMALLOC DDNAME=CIGPUNCH,IOTYPE=W
*
        COPY CIGPUNCH TO FTPIN
        FLMTRNSL FUNCTN=BUILD,CALLNAM='STEP2D: CIGFCOPY',          X
        COMPILE=CIGFCOPY,DSNAME=FLHQ1.FLHQ2.LOADLIB,             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=BUILD,CALLNAM='STEP2A: IRXJCL',            X
        COMPILE=IRXJCL,                                           X
        OPTIONS='CIGRFTP1 MEMBER=@@FLMMBR,PROJECT=@@FLMMPRJ,GROUPX
        =@@FLMGRP,TYPE=@@FLMTYP '
        FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
        FLMCPYLB FLHQ1.FLHQ2.JCLLIB
        FLMALLOC DDNAME=SYSTSPT,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
*****
*
* PROMOTE TRANSLATOR
*
* STEP 1A: GET THE LONGNAME FROM THE SLR
*
*****
        FLMTRNSL FUNCTN=COPY,CALLNAM='STEP1A: CIGFPARM',          X
        COMPILE=CIGFPARM,DSNAME=FLHQ1.FLHQ2.LOADLIB,             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: CIGFCOPY',          X
        COMPILE=CIGFCOPY,DSNAME=FLHQ1.FLHQ2.LOADLIB,             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: C9LSLR',  X
        COMPILE=C9LSLR,DSNAME=FLHQ1.FLHQ2.LOADLIB
        FLMALLOC DDNAME=CIGLOG,IOTYPE=W
        FLMALLOC DDNAME=CIGIN,IOTYPE=U
        FLMALLOC DDNAME=CIGPUNCH,IOTYPE=W
*
        COPY CIGPUNCH TO FTPIN
        FLMTRNSL FUNCTN=COPY,CALLNAM='STEP2D: CIGFCOPY',          X
        COMPILE=CIGFCOPY,DSNAME=FLHQ1.FLHQ2.LOADLIB,             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
              COMPIL=IRXJCL,                                     X
              OPTIONS='CIGRFTP1 MEMBER=@@FLMMBR,PROJECT=@@FLMPRJ,GROUPX
              =@@FLMGRP,TYPE=@@FLMTYP'
          FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
          FLMCPYLB FLHQ1.FLHQ2.JCLLIB
          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

```

23. CIG@FTP1 S-FTP Translator

Review and modify the CIGRFTP1 REXX Script

The following REXX script can be found in the JCLLIB library offloaded during the product install.

Issue a ‘CAPS OFF’ command prior to editing this member!

1. Using ISPF EDIT, access member CIGRFTP1 in the JCLLIB library.
2. Issue a ‘CAPS OFF’ command on the command line of the edit session to ensure case sensitivity.
3. Substitute your site-specific values (identified on the Installation Worksheet.) for those values shown below in bold.

```

/* rexx */
/* ----- */
/* CIGRFTP1 - USED WITH CIG@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

    /* ----- */
    /* 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

/* ----- */
/* Get parms */
/* There will be four parms. */
/* ----- */
GetParms:
  do while (request ~= '')
    parse var request parm', 'request
    parse var parm var1 "=" vall
    interpret var1 "=" vall'
  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)
      pcFileNameexe = substr(pcFileName, 1, wheredot) || 'exe'
    end
  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

```

```

/* ----- */
/*   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 'mbrexex'"
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 "pcFileNameexe member" "mbrexex "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

/* ----- */
/* 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.ftplug')"
"EXECIO * DISKW MSGFILE (STEM input. FINIS"
"free dd(MSGFILE)"

return

```

24. CIGRFTP1 Rexx Script

Step 4: Update Your Project Definition

The purpose of this step is to review the requirements for updating your current project definition to include the new CIG@FTP1 translator. The translator is included in the JCLLIB library. Include this library in your project definition JCL SYSLIB DDNAME so that the compiler can find the language definition member.

The box below shows the statements required to include the new translator. Include these statements in your current project definition JCL and submit.

Example of Project Definition Update

```

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

```

25. Copy statement example

CHECKPOINT #3

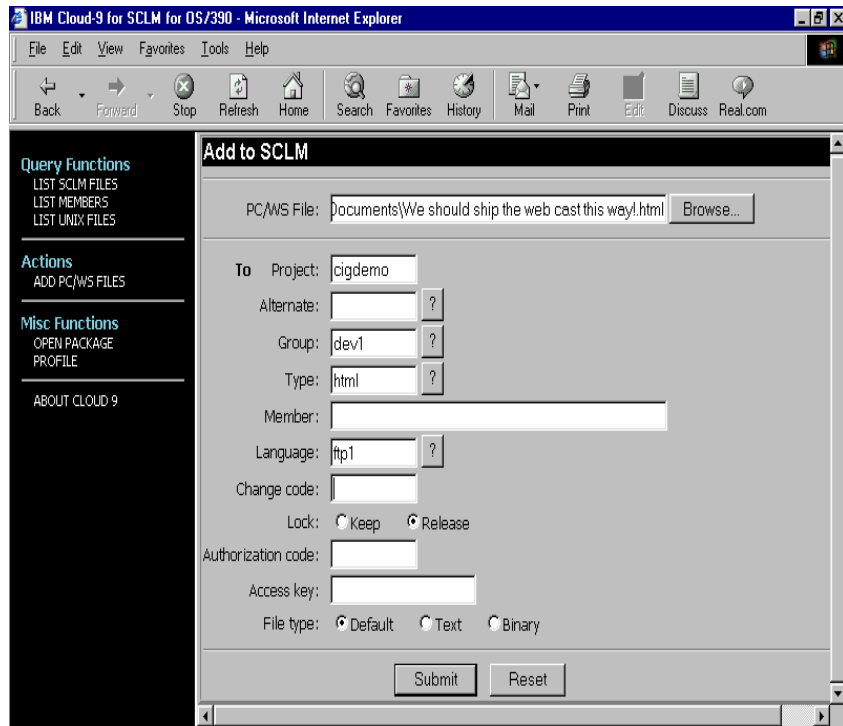
At this point, you should have completed the following tasks:

Task	Completed?
Reviewed and modified translator CIG@FTP1?	
Reviewed and modified REXX script CIGRFTP1?	
Updated your project definition include the CIG@FTP1?	

26. Checkpoint 3

Step 5. Test the S-FTP Translator

1. Add a piece of source defined to the type and language supported. For example, add a type called HTML with a language of FTP1. The Cloud 9 screen shot below shows the process of adding in an HTML file into with a language of FTP1.



27. Example of adding a HTML type, language FTP1

2. Through Cloud 9, request a build of the source to invoke the Build CIG@FTP1 translator. View the batch job output. The following is the BLDMSG 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 *****
```

28. BLDMSG Output

```

READY
ISPSTART CMD(%TEMPNAME)
  3 *- * / * ----- */
  4 *- * / * CIGRFTP1 - USED WITH CIG@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"="vall
104 *- *   interpret var1"="'vall'
   *- *   MEMBER=vall

(More..)

 38 *- * if (group == 'DEV1')
   *- * then
   *- * do
40 *- *   /* ----- */
41 *- *   /* Set Target Location */
42 *- *   /* Example of ftp to a OS/390 Unix System Services location */
43 *- *   /* ----- */
44 *- *   /* ftp -> os/390 */
45 *- *   ipaddr='999.99.999.99'
   *- *   port=21
46 *- *   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

```

29. Rexx Script Trace Data

3. View the 'userid.ftplug' dataset updated in the CIGRFTP1 script. There should be data in the file and it should look roughly like the following. Any real ID's or ip-addresses have been changed to dummy values for security purposes.

```
EZA1736I FTP
EZA1450I CIG 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 CIG 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
```

30. User FTP log example

III. S-JDK SCLM Java Development Kit

Chapter Scope

This chapter contains the Cloud 9 S-JDK prototype translator implementation. The steps in this chapter are organized into four major sections:

- Before you begin
- Customizing Translators and Translator Control Files
- Defining all S-JDK inventory, USS and Cloud 9 pieces
- Perform Installation Verification Procedures (IVP)

These steps should be followed in the order that they are presented. Once you have successfully completed all of the steps and executed the test procedure in Step 8, you will be ready to use the S-JDK.

Warning – Read 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 prior to attaching the SCLM translators.

The SCLM part of this solution is standard SCLM. There are translators, types, languages and control files. The JAVA/USS side of the setup may be foreign to the SCLM administrators so we recommend the following exercises before moving onto setting up the prototype translators.

Other Documents For Reference

There are several Redbooks available from CIG on the issue of USS and JAVA. The following is just the start. The URL to the CIG Redbooks site is www.redbooks.ibm.com. From there you can download a PDF or a link to the Redbooks.

The recommended list of Redbooks is as follows:

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

Verify JAVA/USS Environment

- Verify that you have access to USS environment. Check with your USS Administrator for information on 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.



During this installation, you will modify several JCL members and Unix files. Some of these files contain case-sensitive values. It is imperative that *prior* to modifying the JCL and Unix members, you issue the CAPS OFF command to ensure that automatic upper casing of the Unix members does not occur. For your convenience, the following icon will be placed in each step where case-sensitive Unix values are an issue.



A Step-by-Step Approach

BEFORE YOU BEGIN...	
◆	Review System and Software Requirements
Review Default SCLM Inventory Locations and USS locations	
1.	Review default S-JDK values and determine actual inventory to be used.
2.	Review and modify all translators and translator control file members
Perform SCLM S-JDK Inventory Definitions	
3.	Modify and run CIGTALIB,CIGTAVSM, and CIGTPDEF to build S-JDK Project Definitions
4.	Modify and run CIGC9J06 to define S-JDK types to Cloud 9
5.	Modify and run CIGTAUNX to define USS directories
Test Pilot using CLOCKJ and CLOCKH application	
8.	Add Clock2.java and Clockh.html IVP programs
9.	Invoke Clockh.html to display Time of Day java IVP

31. Enabling the S-JDK

Before You Begin: Review Software and Assumptions

In this step you will review the system and software requirements for enabling the S-JDK.

Assumptions:

This solution guide does not cover configuring the JAVA/USS environment. Enabling and securing the JAVA/USS environment is beyond the scope of the Cloud 9 product and thus this chapter.

This chapter is meant to show the prototypes delivered with the JAVA/USS product. The actual implementation of JAVA/USS in your environment may be more complex, but this is meant as a starting point.

Step 1. 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. Prior to making modifications to these members, please review the worksheets below and fill in the 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 prior to making decisions about the values.

SCLM Inventory Value Worksheet

SCLM Inventory Name	Default Value	Your Value
Project	CIGDEMO	
Alt-project	CIGDEMO	
Group	DEV,QA,REL	
Types	JAVA,JAR,HTML,GRAPHICS	
Languages	JAVA,JAR,EBIZTEXT,EBIZJBIN	

32. SCLM Inventory Value Worksheet

USS Directory Value Worksheet

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

33. USS Directory Value Worksheet

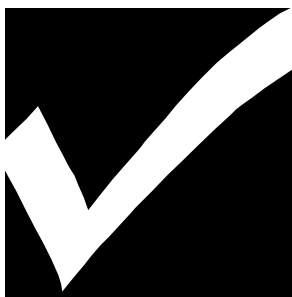
Determining Types Exist?

To determine if a type exists, go to Cloud 9, list SCLM members. For the Group, list the types and verify that the types are there. If they do not exist, then you must define the types and datasets to SCLM. (See Step 3 for further instructions)

The language should not be there yet.

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 CIGC9J06. Review the output from the job, verifying that the SCLM has been defined with the proper attributes. If not, modify this job to define the Types to Cloud 9.



CHECKPOINT #4

At this point the following tasks should have been completed.

Tasks	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.	

35. Checkpoint 4

Step 3. Review and modify all Translators and Control Files

S-JDK Translators:

CIGTJTXT	Translator for HTML Type
CIGTJBIN	Translator for binary Graphic Types
CIGTJAVA	Translator for JAVA
CIGTJAR	Translator for JAR

S-JDK Translator Control Files:

CIGTJCMP	Input to CIGTJAR Translator Compile shell for JAR type
CIGTJMAP	Input to CIGTJAR Translator USS Output Control for JAR type
CIGTJAVC	Input to CIGTJAVA Translator Compile Shell for JAVA type
CIGTUMAP	Input to CIGTJAVA Translator USS Output Control for JAVA type
CIGTCPTH	Input to JAVA and JAR Translators %classpath% Substitution.
CIGTULOC	Input to all S-JDK Translators SCLM to USS mapping rules.

Review and modify CIGTJAVA translator

The following sample translator can be found in the CIG.SCIGJCL dataset. All “review and modify” values will be highlighted in BOLD for easy reading. This translator may not require customization, assuming that the default CIG libraries will be used for input.

```
*-----*
* NAME:      CIGTJAVA                               *
* PURPOSE:   S-JDK TRANSLATOR FOR TYPE = JAVA, LANGUAGE = JAVA. *
*-----*
* 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 *
* CIG.SCIGCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECS. *
*-----*
* YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING *
* CONTROL FILES: *
*-----*
```

```

*          CIG.SCIGCGI (CIGTCPTH)          *
*          CIG.SCIGCGI (CIGTULOC)         *
*          CIG.SCIGCGI (CIGTUMAP)         *
*          CIG.SCIGCGI (CIGTJAVC)         *
*-----*
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)
FLMALLOCC DDNAME=SOURCE, IOTYPE=A
FLMCPYLB @@FLMDSN (@@FLMMBR)
*-----*
*          BUILD TRANSLATOR              *
*-----*
FLMTRNSL FUNCTN=BUILD,                      C
      CALLNAM='INVOKE JAVAC',                C
      COMPILE=IRXJCL,                        C
      OPTIONS='CIGTRJVC @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@C
      FLMMBR'
FLMALLOCC IOTYPE=A, DDNAME=SYSEXEC
FLMCPYLB CIG.SCIGCGI
FLMALLOCC DDNAME=FILEIN, IOTYPE=S, KEYREF=SINC
FLMALLOCC DDNAME=UNIXMAP, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTUMAP)
FLMALLOCC DDNAME=CLASSPTH, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTCPTH)
FLMALLOCC DDNAME=JCOMPILE, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTJAVC)
FLMALLOCC DDNAME=JAVALIST, IOTYPE=O, PRINT=Y, RECNUM=60000,      C
      DFLTTY=JAVALIST, KEYREF=LIST, LRECL=256, LANG=EBIZTEXT
FLMALLOCC DDNAME=WORKFILE, IOTYPE=W, LRECL=32000, RECFM=V
FLMALLOCC DDNAME=SYSPRINT, IOTYPE=O, RECFM=FBA, LRECL=121,      C
      RECNUM=2500, PRINT=N
FLMALLOCC DDNAME=CLASSES, IOTYPE=P, DFLTTY=JAVACLAS,           C
      RECFM=VB, LRECL=256, RECNUM=60000, LANG=EBIZBIN,          C
      KEYREF=OUT1, BLKSIZE=27998
*-----*
*          PROMOTE TRANSLATOR            *
*-----*
FLMTRNSL FUNCTN=COPY,                      C
      CALLNAM='UNIX PROMOTE',                C
      COMPILE=IRXJCL,                        C
      PDSDATA=Y,                              C
      OPTIONS='CIGTRJVP TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTC
      YP @@FLMMBR @@FLMTOG'
FLMALLOCC IOTYPE=A, DDNAME=SYSEXEC
FLMCPYLB CIG.SCIGCGI
FLMALLOCC DDNAME=FILEIN, IOTYPE=A
FLMCPYLB @@FLMDSN (@@FLMMBR)
FLMALLOCC DDNAME=UNIXLOC, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTULOC)
FLMALLOCC DDNAME=WORKFILE, IOTYPE=W, LRECL=32000, RECFM=V
*-----*

```

36. CIGTJAVA – Build and Promote JAVA S-JDK Translator

Review and modify CIGTJAR Translator

The following sample translator can be found in the CIG.SCIGJCL installed with SMP/E. All “review and modify” values will be highlighted in BOLD for easy reading.

```

*-----*
* NAME:      CIGTJAR                               *
* PURPOSE:   S-JDK TRANSLATOR FOR TYPE = JAR, LANGUAGE = JAR. *
*-----*
* 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 *
* CIG.SCIGCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECS. *
*-----*
* YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING *
* CONTROL FILES: *
* CIG.SCIGCGI (CIGTJMAP) *
* CIG.SCIGCGI (CIGTCPTH) *
* CIG.SCIGCGI (CIGTJCMP) *
* CIG.SCIGCGI (CIGTULOC) *
*-----*
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='INVOKE JAR',                 C
                COMPILE=IRXJCL,                       C
                OPTIONS='CIGTRJAR @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTYP @@C
                FLMMBR'
FLMALLOC IOTYPE=A,DDNAME=SYSEXEC
FLMCPYLB CIG.SCIGCGI
FLMALLOC DDNAME=FILEIN,IOTYPE=S,KEYREF=SINC
FLMALLOC DDNAME=JARMAP,IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTJMAP)
FLMALLOC DDNAME=CLASSPTH,IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTCPTH)
FLMALLOC DDNAME=JARCOMP,IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTJCMP)
FLMALLOC DDNAME=JARLIST,IOTYPE=O,PRINT=Y,RECNUM=60000,    C
                DFLTTYP=JARLIST,KEYREF=LIST,LRECL=256,LANG=EBIZTEXT
FLMALLOC DDNAME=JAR,IOTYPE=O,PRINT=Y,RECNUM=60000,      C
                DFLTTYP=JAR,KEYREF=OBJ,LRECL=256,LANG=EBIZBIN
FLMALLOC DDNAME=WORKFILE,IOTYPE=W,LRECL=32000,RECFM=V
FLMALLOC DDNAME=SYSPRINT,IOTYPE=O,RECFM=FBA,LRECL=121,  C
                RECNUM=2500,PRINT=N
FLMALLOC DDNAME=CLASSES,IOTYPE=P,DFLTTYP=JAVACLAS,    C
                RECFM=VB,LRECL=256,RECNUM=60000,LANG=EBIZBIN,
                KEYREF=OUT1,BLKSIZE=27998
*-----*
                PROMOTE TRANSLATOR                    *
*-----*
FLMTRNSL FUNCTN=COPY,                                C
                CALLNAM='UNIX PROMOTE',               C
                COMPILE=IRXJCL,                       C
                PDSDATA=Y,                            C
                OPTIONS='CIGTRJVP BINARY @@FLMPRJ @@FLMALT @@FLMGRP @@FLC
                MTYP @@FLMMBR @@FLMTOG'
FLMALLOC IOTYPE=A,DDNAME=SYSEXEC

```

```

FLMCPYLB CIG.SCIGCGI
FLMALLOD DDNAME=FILEIN, IOTYPE=A
FLMCPYLB @@FLMDSN(@@FLMMBR)
FLMALLOD DDNAME=UNIXLOC, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTULOC)
FLMALLOD DDNAME=WORKFILE, IOTYPE=W, LRECL=32000, RECFM=V
*-----*

```

37. CIGTJAR – Build and Promote JAR S-JDK Translator

Review and modify CIGTJTXT translator

The following sample translator can be found in the CIG.SCIGJCL dataset. All “review and modify” values will be highlighted in BOLD for easy reading.

```

*-----*
* NAME:      CIGTJTXT                               *
* 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 *
* CIG.SCIGCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECs. *
*-----*
* YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING *
* CONTROL FILES: *
* CIG.SCIGCGI (CIGTULOC) *
*-----*
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)
FLMALLOD DDNAME=SOURCE,IOTYPE=A
FLMCPYLB @@FLMDSN(@@FLMMBR)
*-----*
* BUILD TRANSLATOR *
*-----*
FLMTRNSL FUNCTN=BUILD,                               C
              CALLNAM='UNIX BUILD',                   C
              COMPILE=IRXJCL,                         C
              OPTIONS='CIGTRJVP TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTC
              YP @@FLMMBR @@FLMTOG @@FLMBIO'
FLMALLOD IOTYPE=A,DDNAME=SYSEXEC
FLMCPYLB CIG.SCIGCGI
FLMALLOD DDNAME=FILEIN, IOTYPE=S,KEYREF=SINC
FLMALLOD DDNAME=UNIXLOC, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTULOC)
FLMALLOD DDNAME=WORKFILE, IOTYPE=W, LRECL=32000, RECFM=V
*-----*
* PROMOTE TRANSLATOR *
*-----*
FLMTRNSL FUNCTN=COPY,                               C
              CALLNAM='UNIX PROMOTE',                 C
              COMPILE=IRXJCL,                         C
              PDSDATA=Y,                              C
              OPTIONS='CIGTRJVP TEXT @@FLMPRJ @@FLMALT @@FLMGRP @@FLMTC
              YP @@FLMMBR @@FLMTOG'
FLMALLOD IOTYPE=A,DDNAME=SYSEXEC

```

```

FLMCPYLB CIG.SCIGCGI
FLMALLOC DDNAME=FILEIN, IOTYPE=A
FLMCPYLB @@FLMDSN(@@FLMMBR)
FLMALLOC DDNAME=UNIXLOC, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTULOC)
FLMALLOC DDNAME=WORKFILE, IOTYPE=W, LRECL=32000, RECFM=V
*-----*

```

38. CIGTJTXT – Build and Promote JAR S-JDK Translator

Review and modify CIGTJBIN translator

The following sample translator can be found in the CIG.SCIGJCL installed dataset. All “review and modify” values will be highlighted in BOLD for easy reading.

```

*-----*
* NAME: CIGTJBIN *
* 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 *
* CIG.SCIGCGI. YOU DO NOT HAVE TO MODIFY THE REXX EXECS. *
*-----*
* YOU DO HAVE TO REVIEW AND POSSIBLY MODIFY THE FOLLOWING *
* CONTROL FILES: *
* CIG.SCIGCGI (CIGTULOC) *
*-----*
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
COMPILE=IRXJCL, C
OPTIONS='CIGTRJVP BINARY @@FLMPRJ @@FLMALT @@FLMGRP @@FLC
MTYP @@FLMMBR @@FLMTOG'
FLMALLOC IOTYPE=A, DDNAME=SYSEXEC
FLMCPYLB CIG.SCIGCGI
FLMALLOC DDNAME=FILEIN, IOTYPE=S, KEYREF=SINC
FLMALLOC DDNAME=UNIXLOC, IOTYPE=A
FLMCPYLB CIG.SCIGCGI (CIGTULOC)
FLMALLOC DDNAME=WORKFILE, IOTYPE=W, LRECL=32000, RECFM=V
*-----*
* PROMOTE TRANSLATOR *
*-----*
FLMTRNSL FUNCTN=COPY, C
CALLNAM='UNIX PROMOTE', C
COMPILE=IRXJCL, C
PDSDATA=Y, C
OPTIONS='CIGTRJVP BINARY @@FLMPRJ @@FLMALT @@FLMGRP @@FLC
MTYP @@FLMMBR @@FLMTOG'
FLMALLOC IOTYPE=A, DDNAME=SYSEXEC
FLMCPYLB CIG.SCIGCGI
FLMALLOC DDNAME=FILEIN, IOTYPE=A
FLMCPYLB @@FLMDSN(@@FLMMBR)

```

```

FLMALLOC DDNAME=UNIXLOC,IOTYPE=A
FLMCPYLB CIG.SCIGCGI(CIGTULOC)
FLMALLOC DDNAME=WORKFILE,IOTYPE=W,LRECL=32000,RECFM=V
* ----- *

```

39. CIGTJBIN – Build and Promote JAR S-JDK Translator

Review and modify CIGTULOC – Common SCLM to USS Life Cycle Map

CASE SENSITIVITY ALERT!



The following translator control files contain case sensitive data. To ensure case sensitivity is in place, please issue the ‘CAPS OFF’ command on the command line of your ISPF session.

The following member can be found in the CGI file dataset. This member is input to all S-JDK translators. It is used to map the SCLM to USS life cycles locations.

```

* ----- *
* CLOUD 9 JAVA/USS S-JDK COMPONENT *
* ----- *
* NAME: CIGTULOC *
* PURPOSE: SCLM TO UNIX LIFE CYCLE MAPPING RULES. *
* ----- *
* prj,alt,grp,typ          unix location      KEEP or DELETE source
*                               on promote
CIGDEMO,CIGDEMO,DEV,GRAPHICS /u/cigdemo/dev/graphics KEEP
CIGDEMO,CIGDEMO,DEV,HTML    /u/cigdemo/dev          KEEP
CIGDEMO,CIGDEMO,DEV,HTML    /u/cigdemo/dev/html     KEEP
CIGDEMO,CIGDEMO,DEV,JAVA    /u/cigdemo/dev          KEEP
CIGDEMO,CIGDEMO,DEV,JAVACLAS /u/cigdemo/dev/classes  KEEP
CIGDEMO,CIGDEMO,DEV,JAVALIST /u/cigdemo/dev/listings KEEP
*
CIGDEMO,CIGDEMO,QA,GRAPHICS /u/cigdemo/qa/graphics  KEEP
CIGDEMO,CIGDEMO,QA,HTML    /u/cigdemo/qa           KEEP
CIGDEMO,CIGDEMO,QA,HTML    /u/cigdemo/qa/html      KEEP
CIGDEMO,CIGDEMO,QA,JAVA    /u/cigdemo/qa           KEEP
CIGDEMO,CIGDEMO,QA,JAVACLAS /u/cigdemo/qa/classes   KEEP
CIGDEMO,CIGDEMO,QA,JAVALIST /u/cigdemo/qa/listings  KEEP
*
CIGDEMO,CIGDEMO,REL,GRAPHICS /u/cigdemo/rel/graphics KEEP
CIGDEMO,CIGDEMO,REL,HTML    /u/cigdemo/rel          KEEP
CIGDEMO,CIGDEMO,REL,HTML    /u/cigdemo/rel/html     KEEP
CIGDEMO,CIGDEMO,REL,JAVA    /u/cigdemo/rel          KEEP
CIGDEMO,CIGDEMO,REL,JAVACLAS /u/cigdemo/rel/classes  KEEP
CIGDEMO,CIGDEMO,REL,JAVALIST /u/cigdemo/rel/listings KEEP

```



40. CIGTULOC – Common SCLM to USS Life Cycle Map

- Position 1** SCLM Life Cycle location.
- Position 2** Unix System Services Life Cycle location.
- Position 3** Source disposition

This control file is used in the Build, Promote and Delete process.

Modify CIGTCPTH – Common %CLASSPATH% Substitution:

The following member can be found in the CGI file.. 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:      CIGTCPTH                                               *
* PURPOSE:   %CLASSPATH% SUBSTITUTION FILE.                         *
* REFER:     DIRECTLY REFERENCED IN TRANSLATOR DDNAME CLASSPTH     *
*-----*
* prj,alt,gr      classpath concatenation
CIGDEMO,CIGDEMO,DEV /u/cigdemo/dev/classes
CIGDEMO,CIGDEMO,DEV /u/cigdemo/qa/classes
CIGDEMO,CIGDEMO,DEV /u/cigdemo/rel/classes
```



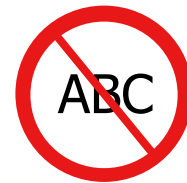
41. CIGTCPTH – Common %CLASSPATH% Substitution

The actual class path may be more complex than one shown above. For instance the core JAVA class libraries may reside in directories outside of the standard SCLM /USS life cycle.

Modify CIGTJAVC – JAVA Compile Shell:

The following member can be found in the CGI file.. This member is input to the CIGTJAVA translator for the JAVA Type. You will need to review the path location with your USS System’s Programmer. The %CLASSPATH% variable is built by Cloud 9 from the values found in the CIGTCPTH member.

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



42. CIGTJAVC Java Compile Shell

Modify CIGTUMAP Java Output Mapping Rules:

The following member can be found in the CGI file. This member is the default input to the CIGTJAVA translator for the JAVA Type and is used to define USS outputs for Java Compiles.

```

* ----- *
* CLOUD 9 JAVA/USS S-JDK COMPONENT *
* ----- *
* NAME: CIGTUMAP *
* PURPOSE: MAPPING RULES FOR JAVA COMPILE USS OUTPUT LOCATIONS. *
* REFER: DIRECTLY REFERENCED IN TRANSLATOR DDNAME UNIXMAP *
* ----- *
*prj,alt,grp java source java class java listing
CIGDEMO,CIGDEMO,DEV,JAVA /u/cigdemo/dev /u/cigdemo/dev/classes /u/cigdemo/dev/listings
CIGDEMO,CIGDEMO,QA,JAVA /u/cigdemo/qa /u/cigdemo/qa/classes /u/cigdemo/qa/listings
CIGDEMO,CIGDEMO,REL,JAVA /u/cigdemo/rel /u/cigdemo/rel/classes /u/cigdemo/rel/listings

```



43. CIGTUMAP – Java Output Mapping Rules

- Position 1** SCLM Java Location
- Position 2** Unix Source for Java Compiler
- Position 3** Unix Target for Compiler Generated Classes
- Position 4** Unix Target for Compiler Generated Listings

Modify CIGTJCMP Jar Compile Shell:

The following member can be found in the CGI file. This member is the default input to the CIGTJAR translator for the JAR Type. You will need to review the path statement with your USS Systems Programmer. The %CLASSPATH% variable is built by Cloud 9 from the values found in the CIGTCPTH member.

```

#-----#
# CLOUD 9 JAVA/USS S-JDK COMPONENT                #
#-----#
# NAME:      CIGTJCMP                             #
# PURPOSE:   JAR COMPILE SHELL                    #
# USAGE:     USED AS INPUT TO THE CIGTJAR TRANSLATOR #
#           READ FROM STANDARD OS/390 PDS DATASET  #
# REFER:     DIRECTLY REFERENCED IN TRANSLATOR DDNAME JARCOMP. #
#-----#
export PATH=/usr/lpp/java/J1.1/bin:$PATH
export CLASSPATH=%CLASSPATH%:$CLASSPATH
cd %CURDIR%
jar cfv %FILENAME% \
%SOURCE%
chmod -fr 777 %FILENAME%
jar tvf %FILENAME%

```



44. CIGTJCMP – JAR Compile Shell

Modify CIGTJMAP - Jar Output Mapping Rules:

The following member can be found in the HTML file. This member is the default input to the CIGTJAR translator for the JAR Type. It is used to define the USS Output locations for Jar compiles.

```

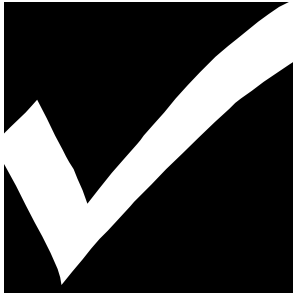
*-----*
* CLOUD 9 JAVA/USS S-JDK COMPONENT                *
*-----*
* NAME:      CIGTJMAP                             *
* PURPOSE:   MAPPING RULES FOR JAR COMPILE USS OUTPUT LOCATIONS. *
* REFER:     DIRECTLY REFERENCED IN TRANSLATOR DDNAME JARMAP     *
*-----*
*prj,alt,grp          jar-cur-dir          jar-exec-loc          jar-listing-loc
CIGDEMO,CIGDEMO,DEV,JAVAMAKE /u/cigdemo/dev/classes /u/cigdemo/dev/jar /u/cigdemo/dev/listings

```



45. CIGTJMAP – JAR Output Mapping Rules

- Position 1** SCLM location for JAR Files
- Position 2** Unix Source for Classes to put into JAR
- Position 3** Unix Target for JAR Compile
- Position 4** Unix Target for JAR Listings



CHECKPOINT #5

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

Task	Completed?
Reviewed and modified JAVA translator CIGTJAVA?	
Reviewed and modified JAR translator CIGTJAR?	
Reviewed and modified TEXT translator CIGTJTXT?	
Reviewed and modified GRAPHICS translator CIGTJBIN?	
Reviewed and modified Classpath control file CIGTCPTH?	
Reviewed and modified Unix JCL Shell CIGTJCIG?	
Reviewed and modified the USS to SCLM mapping control file CIGTULOC?	
Reviewed and modified JAR Output mapping control file CIGTJMAP?	
Reviewed and modified JAVA Output mapping control file CIGTUMAP?	
Reviewed and modified JAVA compile shell CIGTJAVC?	
Reviewed and modified JAR compile shell CIGTJCMP?	

46. Checkpoint 5

Step 4: Update the Project Definition

Project Definition Updates

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 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 will be 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, CIGTPDEF, can be found in the CIG.SCIGJCL library. This is meant as an example. Please review with your SCLM administrator about whether the new types will be an isolated project or part of an existing project.

```
/** (JOB CARD)
/**-----*
/** NAME: CIGTPDEF *
/** 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. *
/** 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=CIG.SCIGJCL,DISP=SHR
// DD DSN=ISP.SISPMACS,DISP=SHR
//SYSPUNCH DD DUMMY
//SYSUT1 DD UNIT=TDISK,SPACE=(TRK,(5,15))
//SYSPRINT DD SYSOUT=*
// PEND
/**-----*
//LINK PROC
//STEP2LNK EXEC PGM=IEWL,REGION=2048K,COND=(0,NE),
// PARM='LIST,XREF,LET,RENT,REUS,CALL'
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=CIGDEMO.PROJDEFS.OBJLIB,DISP=SHR
//SYSLMOD DD DSN=CIGDEMO.PROJDEFS.LOAD,DISP=SHR
//SYSUT1 DD UNIT=TDISK,SPACE=(TRK,(5,15))
/**-----*
// PEND
//COMPILE EXEC COMP
```

```

//SYSIN DD *
TITLE '*** PROJECT DEFINITION FOR PROJECT=CIGDEMO ***'
CIGDEMO FLMABEG
*
* *****
* * DEFINE THE AUTHORIZATION 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
JAVALLIST 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=CIGDEMO.PROJDEFS.ACCT, X
VERS=CIGDEMO.PROJDEFS.VERSION, X
XREF=CIGDEMO.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=JAVALLIST,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=JAVALLIST,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=JAVALLIST,VERSION=YES
FLMATVER GROUP=REL,TYPE=PACKAGES,VERSION=YES
*

```

```

*****
*          LANGUAGE DEFINITION TABLES          *
*****
*          TRANSLATOR          LANGUAGE          *
*          COPY FLM@ARCD          -- ARCHDEF          --
*          COPY FLM@COB2          -- COBOL          --
*          COPY CIGTJAR          -- JAR          --
*          COPY CIGTJAVA          -- JAVA          --
*          COPY CIGTJBIN          -- BINARY E-BUSINESS OBJECTS --
*          COPY CIGTJTXT          -- TEXT E-BUSINESS OBJECTS --
*
*****
          FLMAEND
/*
//SYSLIN DD DSN=CIGDEMO.PROJDEFS.OBJLIB(CIGDEMO),DISP=SHR
//LINK EXEC LINK
//SYSLIN DD *
          INCLUDE SYSLIB(CIGDEMO)
          NAME CIGDEMO(R)
/*

```

47. Sample CIGDEMO Project Definition

Optionally, if you are building a new isolated project for the S-JDK system, then you will need to run two additional jobs: CIGTALIB and CIGTAVSM. These JCL streams will allocate all of the group libraries and the SCLM VSAM files, respectfully.

Step 4a: Allocate New S-JDK Type Datasets

Allocate SCLM Type Files – CIGTALIB

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 will allocate these required libraries.

```

/* (JOB CARD)
/*-----*
/* NAME: CIGTALIB *
/* 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 CIGDEMO.DEV.COBOL
DELETE CIGDEMO.DEV.DOC
DELETE CIGDEMO.DEV.GRAPHICS
DELETE CIGDEMO.DEV.HTML
DELETE CIGDEMO.DEV.JAR
DELETE CIGDEMO.DEV.JAVA
DELETE CIGDEMO.DEV.JAVACLAS
DELETE CIGDEMO.DEV.JAVALIST
DELETE CIGDEMO.DEV.JCL
DELETE CIGDEMO.DEV.PACKAGES
DELETE CIGDEMO.QA.COBOL
DELETE CIGDEMO.QA.DOC
DELETE CIGDEMO.QA.GRAPHICS
DELETE CIGDEMO.QA.HTML
DELETE CIGDEMO.QA.JAR
DELETE CIGDEMO.QA.JAVA
DELETE CIGDEMO.QA.JAVACLAS
DELETE CIGDEMO.QA.JAVALIST
DELETE CIGDEMO.QA.JCL
DELETE CIGDEMO.QA.PACKAGES
DELETE CIGDEMO.REL.COBOL
DELETE CIGDEMO.REL.DOC
DELETE CIGDEMO.REL.GRAPHICS
DELETE CIGDEMO.REL.HTML
DELETE CIGDEMO.REL.JAR
DELETE CIGDEMO.REL.JAVA
DELETE CIGDEMO.REL.JAVACLAS
DELETE CIGDEMO.REL.JAVALIST
DELETE CIGDEMO.REL.JCL
DELETE CIGDEMO.REL.PACKAGES
DELETE CIGDEMO.DEV.COBOL.VERSION
DELETE CIGDEMO.DEV.DOC.VERSION
DELETE CIGDEMO.DEV.GRAPHICS.VERSION
DELETE CIGDEMO.DEV.HTML.VERSION
DELETE CIGDEMO.DEV.JAVA.VERSION
DELETE CIGDEMO.QA.COBOL.VERSION
DELETE CIGDEMO.QA.DOC.VERSION
DELETE CIGDEMO.QA.GRAPHICS.VERSION
DELETE CIGDEMO.QA.HTML.VERSION
DELETE CIGDEMO.QA.JAR.VERSION
DELETE CIGDEMO.QA.JAVA.VERSION
DELETE CIGDEMO.QA.JAVACLAS.VERSION
DELETE CIGDEMO.QA.JAVALIST.VERSION
DELETE CIGDEMO.QA.JCL.VERSION
DELETE CIGDEMO.QA.PACKAGES.VERSION
DELETE CIGDEMO.REL.COBOL.VERSION
DELETE CIGDEMO.REL.DOC.VERSION
DELETE CIGDEMO.REL.GRAPHICS.VERSION
DELETE CIGDEMO.REL.HTML.VERSION
DELETE CIGDEMO.REL.JAR.VERSION
DELETE CIGDEMO.REL.JAVA.VERSION
DELETE CIGDEMO.REL.JAVACLAS.VERSION
DELETE CIGDEMO.REL.JAVALIST.VERSION
DELETE CIGDEMO.REL.JCL.VERSION
DELETE CIGDEMO.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=CIGDEMO.DEV.COBOL
//FILE02 EXEC ALLOC,FILE=CIGDEMO.DEV.DOC
//FILE03 EXEC ALLOC,FILE=CIGDEMO.DEV.GRAPHICS
//FILE04 EXEC ALLOC,FILE=CIGDEMO.DEV.HTML
//FILE05 EXEC ALLOC,FILE=CIGDEMO.DEV.JAR
//FILE06 EXEC ALLOC,FILE=CIGDEMO.DEV.JAVA
//FILE07 EXEC ALLOC,FILE=CIGDEMO.DEV.JAVACLAS
//FILE08 EXEC ALLOC,FILE=CIGDEMO.DEV.JAVALIST
//FILE10 EXEC ALLOC,FILE=CIGDEMO.DEV.JCL
//FILE11 EXEC ALLOC,FILE=CIGDEMO.DEV.PACKAGES
//FILE12 EXEC ALLOC,FILE=CIGDEMO.QA.COBOL
//FILE13 EXEC ALLOC,FILE=CIGDEMO.QA.DOC
//FILE14 EXEC ALLOC,FILE=CIGDEMO.QA.GRAPHICS
//FILE15 EXEC ALLOC,FILE=CIGDEMO.QA.HTML
//FILE16 EXEC ALLOC,FILE=CIGDEMO.QA.JAR
//FILE17 EXEC ALLOC,FILE=CIGDEMO.QA.JAVA
//FILE18 EXEC ALLOC,FILE=CIGDEMO.QA.JAVACLAS
//FILE19 EXEC ALLOC,FILE=CIGDEMO.QA.JAVALIST
//FILE21 EXEC ALLOC,FILE=CIGDEMO.QA.JCL
//FILE22 EXEC ALLOC,FILE=CIGDEMO.QA.PACKAGES
//FILE23 EXEC ALLOC,FILE=CIGDEMO.REL.COBOL
//FILE24 EXEC ALLOC,FILE=CIGDEMO.REL.DOC
//FILE25 EXEC ALLOC,FILE=CIGDEMO.REL.GRAPHICS
//FILE26 EXEC ALLOC,FILE=CIGDEMO.REL.HTML
//FILE27 EXEC ALLOC,FILE=CIGDEMO.REL.JAR
//FILE28 EXEC ALLOC,FILE=CIGDEMO.REL.JAVA
//FILE29 EXEC ALLOC,FILE=CIGDEMO.REL.JAVACLAS
//FILE30 EXEC ALLOC,FILE=CIGDEMO.REL.JAVALIST
//FILE32 EXEC ALLOC,FILE=CIGDEMO.REL.JCL
//FILE33 EXEC ALLOC,FILE=CIGDEMO.REL.PACKAGES
//*
//FILE50 EXEC VALLOC,FILE=CIGDEMO.DEV.COBOL.VERSION
//FILE51 EXEC VALLOC,FILE=CIGDEMO.DEV.DOC.VERSION
//FILE52 EXEC VALLOC,FILE=CIGDEMO.DEV.GRAPHICS.VERSION
//FILE53 EXEC VALLOC,FILE=CIGDEMO.DEV.HTML.VERSION
//FILE54 EXEC VALLOC,FILE=CIGDEMO.DEV.JAVA.VERSION
//FILE55 EXEC VALLOC,FILE=CIGDEMO.QA.COBOL.VERSION
//FILE56 EXEC VALLOC,FILE=CIGDEMO.QA.DOC.VERSION
//FILE57 EXEC VALLOC,FILE=CIGDEMO.QA.GRAPHICS.VERSION
//FILE58 EXEC VALLOC,FILE=CIGDEMO.QA.HTML.VERSION
//FILE59 EXEC VALLOC,FILE=CIGDEMO.QA.JAR.VERSION
//FILE60 EXEC VALLOC,FILE=CIGDEMO.QA.JAVA.VERSION
//FILE61 EXEC VALLOC,FILE=CIGDEMO.QA.JAVACLAS.VERSION
//FILE62 EXEC VALLOC,FILE=CIGDEMO.QA.JAVALIST.VERSION
//FILE64 EXEC VALLOC,FILE=CIGDEMO.QA.JCL.VERSION
//FILE65 EXEC VALLOC,FILE=CIGDEMO.QA.PACKAGES.VERSION
//FILE66 EXEC VALLOC,FILE=CIGDEMO.REL.COBOL.VERSION
//FILE67 EXEC VALLOC,FILE=CIGDEMO.REL.DOC.VERSION
//FILE68 EXEC VALLOC,FILE=CIGDEMO.REL.GRAPHICS.VERSION
//FILE69 EXEC VALLOC,FILE=CIGDEMO.REL.HTML.VERSION
//FILE70 EXEC VALLOC,FILE=CIGDEMO.REL.JAR.VERSION
//FILE71 EXEC VALLOC,FILE=CIGDEMO.REL.JAVA.VERSION
//FILE72 EXEC VALLOC,FILE=CIGDEMO.REL.JAVACLAS.VERSION
//FILE73 EXEC VALLOC,FILE=CIGDEMO.REL.JAVALIST.VERSION
//FILE75 EXEC VALLOC,FILE=CIGDEMO.REL.JCL.VERSION

```

Step 4b: Optionally, Allocate New Project VSAM Files

Allocate Project VSAM Files - CIGTAVSM

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 CIG.SCIGJCL library, will allocate these required libraries.

```
/** (JOB CARD)
/**-----*
/** NAME: CIGTAVSM *
/** 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 *
/** PERMANENT FILES. *
/** 5) CHANGE THE "DVOLSER" VARIABLE TO THE APPROPRIATE *
/** VOLUME FOR VSAM FILES. *
/**-----*
/** CIGDEMO.PROJDEFS.JCLLIB(ALLOVSAM)
/**-----*
//ACCT1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
DELETE CIGDEMO.PROJDEFS.ACCT
DEFINE CLUSTER +
(NAME('CIGDEMO.PROJDEFS.ACCT') +
CYLINDERS(1 1) +
VOLUMES(DVOLSER) +
KEYS(26 0) +
IMBED +
RECORDSIZE(264 32000) +
SHAREOPTIONS(4,3) +
SPEED +
SPANNED +
UNIQUE) +
INDEX(NAME('CIGDEMO.PROJDEFS.ACCT.I') -
) +
DATA(NAME('CIGDEMO.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=CIGDEMO.PROJDEFS.ACCT,DISP=SHR
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
    REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*****
//* A JOB STEP IS THEN EXECUTED TO INITIALIZE THE FILE.
//*****
//VERS1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
    DELETE CIGDEMO.PROJDEFS.VERSION
    DEFINE CLUSTER +
        (NAME('CIGDEMO.PROJDEFS.VERSION') +
        CYLINDERS(1 1) +
        VOLUMES(DVOLSER) +
        KEYS(40 0) +
        IMBED +
        RECORDSIZE(264 32000) +
        SHAREOPTIONS(4,3) +
        SPEED +
        SPANNED +
        UNIQUE) +
        INDEX(NAME('CIGDEMO.PROJDEFS.VERSION.I') -
        ) +
        DATA(NAME('CIGDEMO.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=CIGDEMO.PROJDEFS.VERSION,DISP=SHR
//SYSPRINT DD SYSOUT=H
//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=CIGDEMO.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=H
//SYSIN DD *
    DELETE CIGDEMO.PROJDEFS.XREF
    DEFINE CLUSTER +
        (NAME('CIGDEMO.PROJDEFS.XREF') +
        CYLINDERS(2 1) +
        VOLUMES(DVOLSER) +
        KEYS(128 0) +
        IMBED +
        RECORDSIZE(264 32000) +

```

```

        SHAREOPTIONS(4,3) +
        SPEED +
        SPANNED +
        UNIQUE) +
        INDEX(NAME('CIGDEMO.PROJDEFS.XREF.I') -
        ) +
        DATA(NAME('CIGDEMO.PROJDEFS.XREF.D') -
        CISZ(2048) +
        FRESpace(50 50) +
        )
/*
//*****
//* INITIALIZE THE CROSS-REFERENCE FILE
//*****
//XREF2      EXEC PGM=IDCAMS
//INPUT DD DSN=&&PARMFILE,DISP=(OLD,DELETE)
//OUTPUT DD DSN=CIGDEMO.PROJDEFS.XREF,DISP=SHR
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
        REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*

```

49. CIGTAVSM – Allocate Project VSAM Files

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 example below shows the default S-JDK definitions required to define the S-JDK types to the Cloud 9 SLR.

Modify and Submit CIGC9J06

1. Using ISPF EDIT, access member CIGC9J06 CIG.SCIGJCL library.
2. This JCL should already have been modified during the base install and IVP process.
3. Update the member including the following SLR definitions (**in BOLD**)
4. Submit the job.

Note that this job should terminate with COND CODE=0.

```
//* (JOB CARD)
//* -----*
//*          CLOUD 9 JAVA/S-JDK COMPONENTS.          *
//* -----*
//* NAME:      CIGC9J06                               *
//* 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=CIGSLR
//STEPLIB   DD DSN=CIG.SCIGLOAD,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=*
```

50. CIGC9J06 – Define S-JDK Types to Cloud 9 SLR

Step 6: Run CIGJAUNX to build S-JDK USS Directories

The purpose of this step is to define the S-JDK Unix Directories. The member is shown with default values.

Modify and Submit CIGJAUNX

1. Using ISPF EDIT, access member CIGJAUNX CIG.SCIGJCL.
2. Issues 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 shown below in bold.
4. Submit the job.

Note that this job should terminate with COND CODE=0.

```
/** (JOB CARD)
/**-----*
/** NAME:      CIGJAUNX                               *
/** 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 CIGdemo */
oshell rm -r /u/cigdemo/dev
oshell rm -r /u/cigdemo/qa
oshell rm -r /u/cigdemo/rel

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

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

```

oshell rmdir -p /u/cigdemo/qa/listings

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

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

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

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

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

/* copy sample files to target location */
oput 'CIG.SCIGHTML(CIGHCLCK)' '/u/cigdemo/dev/clock.html'
oput 'CIG.SCIGHTML(CIGJCLCK)' '/u/cigdemo/dev/Clock2.java'

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

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

```

51. CIGTAUNX – Create USS S-JDK Directories

Step 7: Review CIGJCIG 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 CIGJCIG JCL shell that comes with the base Cloud 9 product.

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

```
)DOT
%JOB CARD%
)ENDDOT
//* ----- *
//* NAME:      CIGJCIG *
//* PURPOSE:   CLOUD 9 FOR SCLM. *
//*           SCLM BATCH SKELETON. *
//* ----- *
//*
//* REQUIRED JCL MODIFICATION: *
//* 1) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET. *
//*    - ISPFQUAL *
//*    - TDISK *
//* ----- *
//* /ROOTDIR/CLOUD9/JCL/CIGJCIG *
//* SCLM-SHELL1 AND FLHQ2 *
//* ----- *
)IF ACTION=MIGRATE
//COPY EXEC PGM=IKJEFT01
)DOT
%COPYFILES%
)ENDDOT
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
)DOT
%OCOPYSYNTAX%
)ENDDOT
/*
)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
```

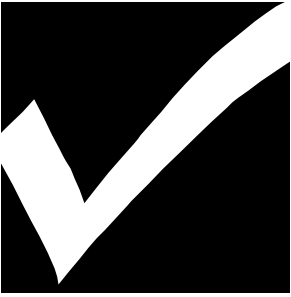
```

//*****
//TSO EXEC PGM=IKJEFT01,REGION=4096K,TIME=1439,DYNAMNBR=200
//SYSTSIN DD *
  ISPSTART CMD(%TEMPNAME)
//SYSTSPRT DD SYSOUT=(*)
//SYSPROC DD DSN=&&CLIST,DISP=(OLD,DELETE)
//*****
//* ISPF LIBRARIES
//*****
//ISPLIB DD DSN=ISPFQUAL.SISPMENU,DISP=SHR
//ISPSLIB DD DSN=ISPFQUAL.SISPSENU,DISP=SHR
// DD DSN=ISPFQUAL.SISPSLIB,DISP=SHR
//ISPLIB DD DSN=ISPFQUAL.SISPPENU,DISP=SHR
//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
//*****
//FLMMSG DD SYSOUT=(*)
)IF ACTION=BUILD
//BLDMSG 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)
)ENDIF
)IF ACTION=PROMOTE
//PROMMSG 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)
//PROMERR 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)
//U2MSG DD SYSOUT=*, MIGRATE
// DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
)IF ACTION=DELETE
//U9LSTS DD SYSOUT=*, DELETE
// DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//U9MSG DD SYSOUT=*, DELETE
// DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//U9RPTS DD DSN=&&DELLIST,DISP=(NEW,PASS), DELETE
// SPACE=(TRK,(5,10),RLSE),
// DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
)ENDIF
)IF ACTION=VERRECOV
//DBUMSG DD SYSOUT=*, VERRECOV
// DCB=(LRECL=80,BLKSIZE=80,RECFM=F)

```

```
)ENDIF
/*-----
) IF ACTION=DELETE
//DELETE EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=200
//STEPLIB DD DSN=CIG.SCIGLOAD,DISP=SHR
//SYSTSIN DD *
    EX 'CIG.SCIGCGI (CIGTRJDL) '
//SYSTSPRT DD SYSOUT=*
//DELLIST DD DSN=&&DELLIST,DISP=(OLD,PASS)
//LISTOUT DD SYSOUT=*,
//          DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//UNIXLOC DD DSN=CIG.SCIGCGI (CIGTULOC) ,DISP=SHR
)ENDIF
```

52. CIGJCIG Unix JCL Shell



CHECKPOINT #6

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

Task	Completed?
Update your project definition with JDK Types and Translators (Optionally CIGTPDEF)	
Allocate New SCLM Type Libraries CIGTALIB.	
Optionally allocate new SCLM project VSAM, only if creating a new project using CIGTAVSM.	
Modified CIGJCIG Cloud 9 JCL shell.	
Run CIGC9J06 to define S-JDK types to Cloud 9	
Run CIGJAUNX to define USS directories for S-JDK application	

53. Checkpoint 6

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 CIGJCLCK and one is an HTML invocation file call CIGHCLCK. There are all delivered in the CIG.SCIGHTML libraries. The CIGTAUNX JCL stream already copied and renamed the files to the JAVA application area as shown below

Member	Already saved into USS as the following:
CIGJCLCK	/cigdemo/dev/Clock2.java
CIGHCLCK	/cigdemo/dev/clock.html

The following steps will allow you to verify the S-JDK translators.

1. Invoke Cloud 9
2. List Unix Files for the /u/cigdemo/dev directories
3. Select Clock2.java from list
4. Migrate Clock2.java into Cloud 9
 - a. Add as a type JAVA
 - b. You should see a short name generated
5. List SCLM files for type JAVA
6. Force a 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. The following is what you should see in the CLASSES directory:

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

54. Directory Entry after Java Compile

Java Listing Example:

The following is what you should 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
```

55. Listing example from USS JAVA Compile

Java SCLM Build Map Example

The following is 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 *****
```

56. Build Map List Example

Step 9: Invoking the Compiled Java

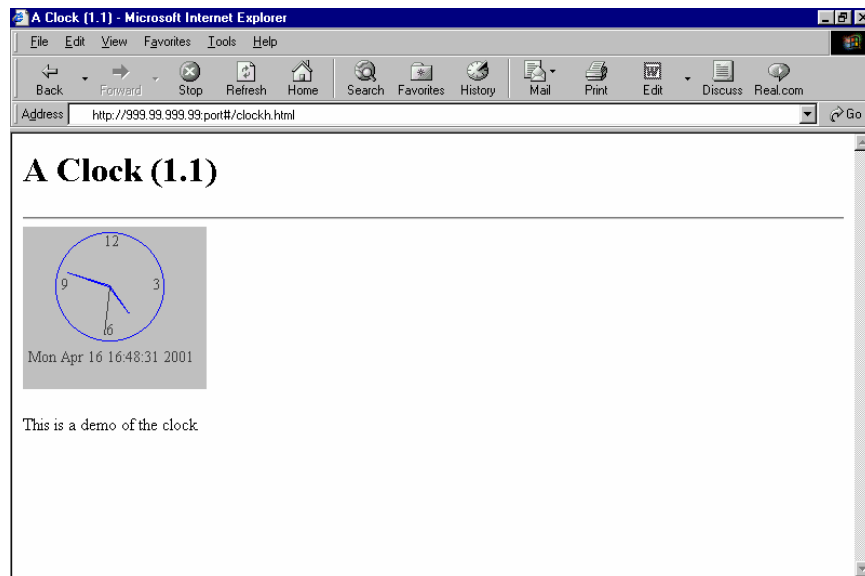
Copy the Clock2.class and the clock.html to a known location on a Web Server. For example, clock.html to the WebSphere root directory and copy the Clock2.class to WebSphere root directory/classes/ directory. You should be able to invoke the compiled Java code by entering the same ip-address and port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

Enter the following in the location area of your browser:

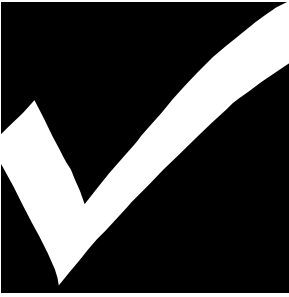
Ip-address:port/clock.html

Result of Clock2.java Compile:

The result should be as follows:



57. Clock2.class Display



CHECKPOINT #7

At this point you should have completed the following tasks:

Task	Completed?
Copied CIGJCLCK to the Cloud 9 to a USS or Desk top location as Clock2.java?	
Added Clock2.java as a JAVA element into SCLM via Cloud 9?	
Reviewed translator output?	
Reviewed population of USS Output directories?	
Copied Clock2.class to the WebSphere /rootdir/classes/ directory?	
Invoked clockh.html from a web server location?	

58. Checkpoint 7

Chapter 6: Cloud 9 Exits

There is one exit point in Cloud 9 that may need to be the modified.

CLZREX00 – Cloud 9 Temporary Dataset Prefix Setting

CLZREX00 is a REXX CGI module that directs Cloud 9 as the high level qualifier to use for temporary datasets. If you do not modify this member, then the default is the userid. The figure below shows the default CLZREX00 delivered on the installation cartridge and copied into the USS directory /rootdir/cgi-bin/clzrex00. Please review this source and modify if needed.

```
/* rexx -----  
- Program: CLZREX00 -  
- Purpose: This program will return a dataset prefix. -  
- The prefix can be 1-16 characters long. -  
- For example: ABC.STEVE -  
- -  
- input parameter: userid -  
- return value...: prefix -  
- -  
----- */  
parse arg uid  
  
/* Example: */  
/* uid = 'CLOUD9.'uid */  
  
return uid  
/* ----- */
```

59. CLZREX00 Sample Exit

Appendix A - Type Definition Worksheet

Type	Language	Defined to SCLM	Defined to Cloud 9 SLR	Case Sensitive Yes/No	Binary Yes/No	Lrecl	Recfm
Java	JAVA				No	256	VB
Jar	JAR				Yes	256	VB
Html	EBIZTEXT				No	256	VB
Graphics	EBIZJBIN				Yes	256	VB

60. Type Definition Matrix

Appendix B - Application Life Cycle Worksheet

Application Life Cycle Requirements

The following worksheet can be used to help determine the needs of each object type at each location in the life cycle. This worksheet can then be used to fully scope out the application implementation.

Host Type	Host Life Cycle Location	Version Control	Production Lock Down	FTP Deploy	USS Builds	Remote Builds * custom scripts req'd	Life Cycle Promotion
.java	Dev	Yes					Yes
.clas	Dev			Yes	Yes		
.html	Dev	Yes	Yes	Yes			Yes

61. Application Life Cycle Worksheet

Table of Figures

1. CIGSOJO2.....	11
2. Example of ADDTYPE Entries.....	13
3. Folder Options.....	14
4. Edit File Type.....	15
5. Netscape Folder Options.....	16
6. CIG SUITE Utility C9LSLR.....	18
7. SLR Long Name Rule Syntax for SCLM.....	19
8. SLR Long Name Rules Parameter Description.....	19
9. SLR Definition Rule Syntax Example.....	20
10. SLR Short Name Entry Syntax.....	20
11. SLR Short Name Rules Parameter Description.....	20
12. SLR Short Name Syntax Example.....	21
13. List Long Name Output.....	21
14. SLR Long Name Entry Syntax.....	21
15. SLR Long Name Rules Parameter Description.....	21
16. SLR Long Name Syntax Example.....	21
17. List Long Name Output.....	21
18. Enabling the S-FTP.....	24
19. S-FTP Value Worksheet.....	27
20. Worksheet for Deploy and Build Target Locations.....	28
21. Type Review Matrix.....	29
22. Checkpoint 2.....	30
23. CIG@FTP1 S-FTP Translator.....	33
24. CIGRFTP1 Rexx Script.....	37
25. Copy statement example.....	37
26. Checkpoint 3.....	38
27. Example of adding a HTML type, language FTP1.....	39
28. BLDMSGs Output.....	39
29. Rexx Script Trace Data.....	41
30. User FTP log example.....	42
31. Enabling the S-JDK.....	45
32. SCLM Inventory Value Worksheet.....	47
33. USS Directory Value Worksheet.....	47
34. Type Review Matix.....	48
35. Checkpoint 4.....	50
36. CIGTJAVA – Build and Promote JAVA S-JDK Translator.....	52
37. CIGTJAR – Build and Promote JAR S-JDK Translator.....	54
38. CIGTJTXT – Build and Promote JAR S-JDK Translator.....	55
39. CIGTJBIN – Build and Promote JAR S-JDK Translator.....	55
40. CIGTULOC – Common SCLM to USS Life Cycle Map.....	56
41. CIGTCPTH – Common %CLASSPATH% Substitution.....	57
42. CIGTJAVC Java Compile Shell.....	57
43. CIGTUMAP – Java Output Mapping Rules.....	58
44. CIGTJCOMP – JAR Compile Shell.....	59
45. CIGTJMAP – JAR Output Mapping Rules.....	59
46. Checkpoint 5.....	60
47. Sample CIGDEMO Project Definition.....	63
48. CIGTALIB SCLM Type Dataset Allocations.....	66
49. CIGTAVSM – Allocate Project VSAM Files.....	68
50. CIGC9J06 – Define S-JDK Types to Cloud 9 SLR.....	69
51. CIGTAUNX – Create USS S-JDK Directories.....	71
52. CIGJCIG Unix JCL Shell.....	74

53. Checkpoint 6	75
54. Directory Entry after Java Compile	76
55. Listing example from USS JAVA Compile	77
56. Build Map List Example	77
57. Clock2.class Display	78
58. Checkpoint 7	79
59. CLZREX00 Sample Exit	80
60. Type Definition Matrix	81
61. Application Life Cycle Worksheet	82
62. FTP Worksheet for Deploy and Build Target Locations	83

INDEX:

BLDMSGs Output.....	39	delete	72
Case Sensitive	29, 48, 81	Delete	56, 72
Classpath	57	Example.....	34, 37, 39, 40
CLZ@FTP1	24, 26, 33, 37, 38, 39, 40	FTP server	28, 83
CLZJIBM JCL shell.....	72	Ip-address	26, 28, 83
CLZREX00 – Cloud 9 Temporary Dataset Prefix Setting.....	80	JAR.....	47, 59, 69
CLZRFTP1 ...	24, 25, 26, 27, 33, 37, 38, 40, 42	JAR Compile Shell.....	59
CLZTCPTH	51, 52, 53, 57, 59, 60	JAR Output Mapping Rules	59
CLZTJAR	51, 53, 54, 55, 56, 59, 60, 63	JAVA.....	47, 57, 58, 69, 76, 77, 79
CLZTJAVA	51, 52, 57, 58, 60, 63	Java Output Mapping Rules	57, 58
CLZTJAVC	51, 52, 57, 60	JCL	23, 24, 44, 47, 69
CLZTJBIN	51, 55, 60, 63	Language is FTP1	29, 48, 81
CLZTJCMP	51, 53, 59, 60	Platform Type	28, 83
CLZTJMAP	51, 53, 59, 60	Redbooks.....	44
CLZTJTXT	51, 54, 60, 63	REXX Script	24, 33
CLZTULOC ..	51, 52, 53, 54, 55, 56, 60, 74	Rexx Script Trace.....	41
CLZTUMAP	51, 52, 57, 58, 60	S-FTP translator	23
Create USS E-JDK Directories	71	S-JDK Type Matrix	50
Define E-JDK Types to Cloud 9 SLR.....	69	Target Platform	28
		Unix.....	44, 70