
Chicago Interface Group, Inc.

Cloud 9 for Endeavor Administration Guide

V 7.0

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

Phone: (773) 524-0998
Fax: (773) 525-6098
Email: support@cigi.net
Web: www.cigi.net

Cloud 9 version 7.0

Cloud 9 is a trademark of Chicago Interface Group, Inc.
CA-Endevor is a registered trademark of Computer Associates International, Inc.

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

Documentation Version April 3, 2006

CONTENTS

Chapter 1: Getting Started	5
Introduction	5
Who should use this manual?	5
How to use this manual?	5
Seeing the Big Picture	6
Required Tasks for all Cross Platform Applications	6
Optional Tasks for a Cross Platform Application:	7
CHAPTER 2: STANDARD CROSS PLATFORM SETUP ...	9
Step 1: Define File Types to Endeavor.....	9
Endeavor Base File Requirements:.....	9
Endeavor Delta File Requirements:.....	10
Required Endeavor Type Attributes Example:.....	10
Step 2: Define the Type to SLR	10
Step 3: Add Type Extension to the OS/390 HTTP Rules File (httpd.conf)..	12
Step 4: Update YOUR Browser's File Type Settings	14
Internet Explorer MIME Setup:	14
CHAPTER 3: THE SLR UTILITY	17
Chapter Scope.....	17
Long Name Support	17
The JCL for C9LSLR	18
The Utility – C9LSLR	19
The Syntax for Defining Types to the SLR	19
<i>Example of SLR Definition Syntax:</i>	<i>20</i>
The Syntax for Adding, Deleting, and Listing Entries in the SLR.....	20
<i>Short Name Syntax:.....</i>	<i>20</i>
<i>Example of SLR Short Name Syntax:</i>	<i>21</i>
<i>List Short Name CIGPUNCH Example:</i>	<i>21</i>
<i>Long Name Syntax:</i>	<i>21</i>

<i>Example of SLR Long Name Syntax:</i>	22
<i>List Long Name CIGPUNCH Example:</i>	22
SLR in Endeavor Processors	23
Using C9LSLR with Endeavor	23
CHAPTER 4: EXITS, HTTPD SECURITY ISSUES, AND CUSTOMIZATIONS	24
CLZREX00 – Cloud 9 Temporary Dataset Prefix Setting.....	24
CLZREXIT – C1UXSITE Support.....	25
Alternate ADDTYPE Definitions Location.....	26
LSERV SUPPORT	28
APPENDIX A - TYPE DEFINITION WORKSHEET	29
APPENDIX B - APPLICATION LIFE CYCLE WORKSHEET 30	
Application Life Cycle Requirements	30
APPENDIX C - FTP DEPLOYMENT WORKSHEET	31

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 and Breeze. It assumes basic knowledge of HTTP server concepts, knowledge of Endeavor 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.

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 the process up into smaller buckets. Break up the process into three buckets; 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 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. Next, you would build the Endeavor Types and allocate the files as per CIG's recommendations. It is highly 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:

This is where the fun really begins. 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 processor to deploy an executable or html file outside of Endeavor? Can you accomplish what you need by implementing and customizing the FTP processors included in the SCM Suite? See Chapter 4 for more information on the E-FTP processors.
2. Will you be compiling Java on the host using USS? Do you want absolute correlation between source and executable, much like standard Endeavor component list relationships? Please review Chapter 5 for more information on the Java/USS processors.
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. The Suite provides a E-FTP processor 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. Endeavor processors that can manage and deploy cross platform objects is discussed later in this document.

Chapter 2: Standard Cross Platform Setup

Regardless of which cross platform applications you chose to support with the SCM Suite, 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 an 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 Endeavor

1. Determine the type name. We recommend 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 Endeavor Base and Delta files per Cloud 9 requirements.

Note that you may need to create separate base and delta files to manage the cross platform, long record and long name type files. The figure below shows the recommended attributes of the base and delta files. If your current base and delta files do not meet these requirements, you will need to allocate base and delta files for the new cross platform types that meet these needs.

Endeavor Base File Requirements:

```
REVERSE  
LRECL=512  
BLKSIZE=23478  
RECFM=VB
```

Endevor Delta File Requirements:

REVERSE
LRECL=23400
BLKSIZE=23478
RECFM=VB

3. Define the type to Endevor using Batch Admin. 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 delta files as defined in this section.

Required Endevor Type Attributes Example:

```
*  
* TYPE = DOC WORD FOR WINDOWS FILES  
*  
DEFINE TYPE DOC  
  TO ENVIRONMENT TEST  
  SYSTEM CIG  
  STAGE NUMBER 1  
  DESCRIPTION "DMV WORD FOR WINDOWS"  
  SOURCE ELEMENT LENGTH 500  
  COMPARE COLUMN 1 TO 500  
  LANGUAGE TEXT  
  DEFAULT PROCESSOR GROUP IS '*NOPROC*'  
  BASE LIBRARY 'CLOUD9.EBASE1.REVERSE'  
  DELTA LIBRARY 'CLOUD9.EDELTA1.REVERSE'  
  DO NOT COMPRESS BASE  
  ELEMENT DELTA FORMAT IS REVERSE  
.
```

Example of Endevor Batch SCL for non-OS/390 Type

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)
/**
*****
/**
/** CIGV2IVP - THE PURPOSE OF THIS JCL IS TO RUN THE SLR IVP.
/** STEP 1 WILL PRINT THE CIGINI DEFINITIONS.
/** STEP 2 WILL LIST IVP SLR RULE DEFINITIONS.
/** STEP 3 WILL ADD ENDEVOR
/** TYPE DEFINITIONS AND THEN LIST ALL RULES
/** IN THE DATABASE.
/** NOTE: - SEE THE CLOUD 9 V7.0 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.
*****
/**
/** 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.
/*
*****
/** STEP 3: ADD DATASET AND TYPE DEFINITIONS TO SLR DATABASE.
/** USE AS IS OR TAILOR WITH LOCAL VALUES.
/**
*****
//STEP3 EXEC PGM=C9LSLR
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGPUNCH DD SYSOUT=*
//CIGLOG DD SYSOUT=*
//CIGIN DD *
ADD NAME RULE FOR ENDEVOR TYPE HTML CASE SENSITIVE.
ADD NAME RULE FOR ENDEVOR TYPE JAVA CASE SENSITIVE.
ADD NAME RULE FOR ENDEVOR TYPE UNIXMAKE CASE SENSITIVE .
ADD NAME RULE FOR ENDEVOR TYPE DOC CASE INSENSITIVE .
LIST NAME RULES.
/*
```

CIGV2IVP

Step 3: Add Type Extension to the OS/390 HTTP Rules File (httpd.conf)

Check the httpd.conf file to see if the file extension you're 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 moviepalayer
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

```

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

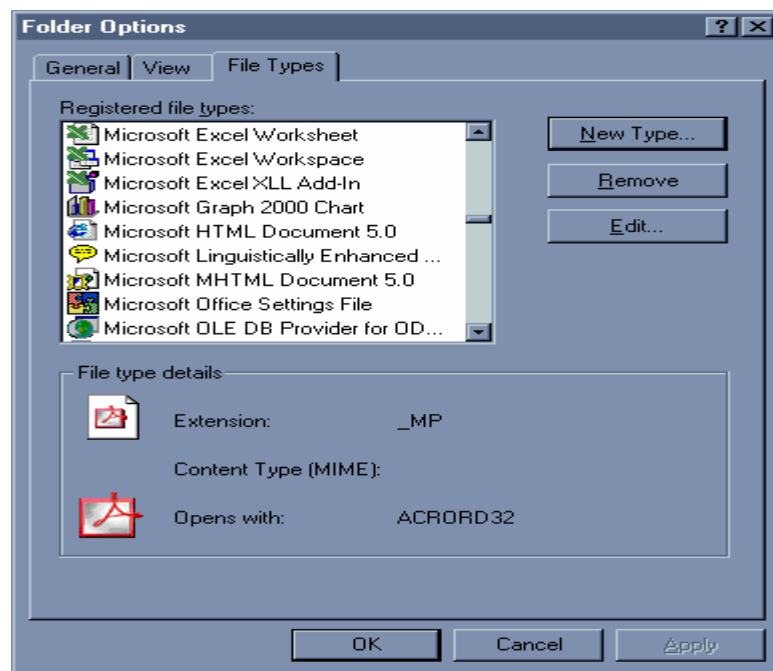
For example if adding an *MS-Excel* file type, the following format would be used:

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

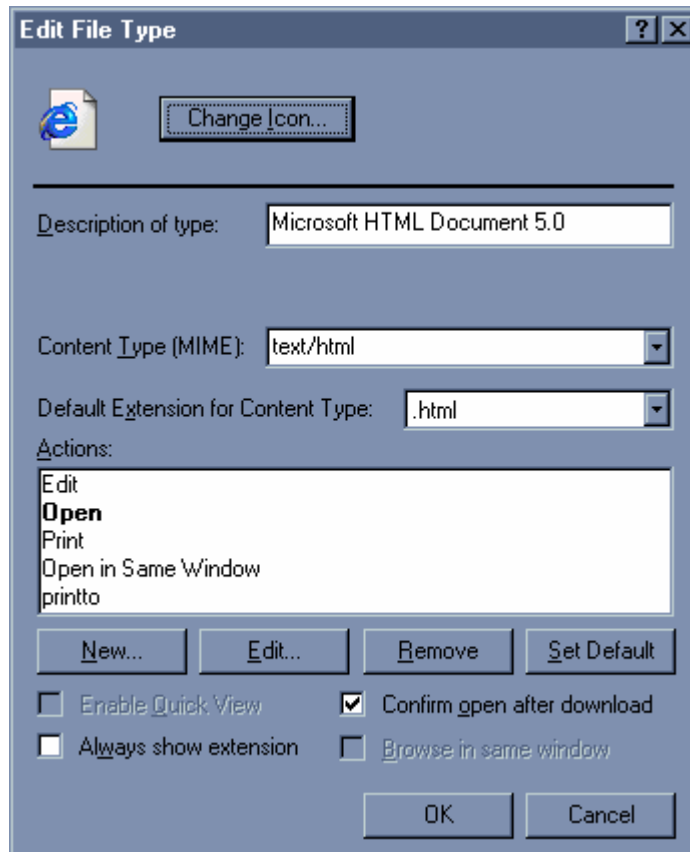
Step 4: Update YOUR Browser's File Type Settings

Internet Explorer MIME Setup:

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



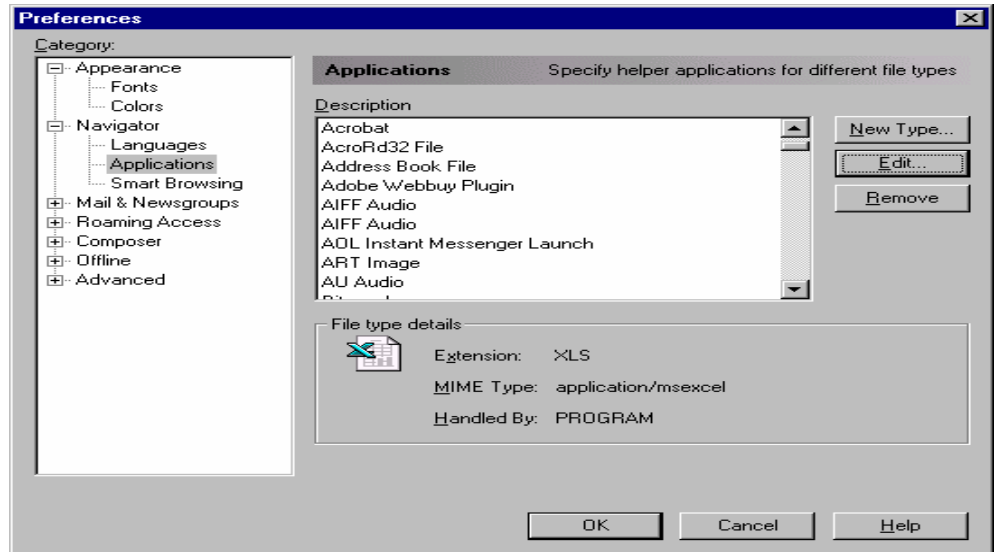
1. Check the list of file types for the file type you will be downloading. If the file type you're looking for is there, then the application currently set to open the file will be displayed. If the file type is there, but set to the wrong application, then select Edit.



2. 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.
3. If the file type your looking for is not in the file list then click the “New Type” button from the Folder Options screen
4. 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



1. 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.
2. Ensure that the correct application is set up to open your file. If it is not set to the right application then select "Edit".
3. 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.
4. 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.
5. 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

This chapter deals with managing cross-platform objects. It explains:

- How long name support works
- The SLR database
- How to define types to Endeavor
- Optimizing your browser settings
- Managing Cross Platform Packages with Breeze

Long Name Support

A key component 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 SCM types are monitored and managed by Cloud 9. Given that an Endeavor based 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 the Cloud 9 to take control. This method is the SLR (short-to-long name registry).

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

The JCL for C9LSLR

The following is the JCL used to define the SLR long name rules to the SCM Suite. It can be found in the Cloud 9 JCLLIB offloaded from the installation tape.

```
/***(JOB CARD)
/**
*****
/**
/** CIGV2IVP - THE PURPOSE OF THIS JCL IS TO RUN THE SLR IVP.
/**          STEP 1 WILL PRINT THE CIGINI DEFINITIONS.
/**          STEP 2 WILL LIST IVP SLR RULE DEFINITIONS.
/**          STEP 3 WILL ADD ENDEVOR
/**          TYPE DEFINITIONS AND THEN LIST ALL RULES
/**          IN THE DATABASE.
/** NOTE:    - SEE THE CLOUD 9 V7.0 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.
*****
/**
/** REQUIRED JCL MODIFICATION:
/** 1) INCLUDE A JOBCARD
/** 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.
/*
*****
/**
/** STEP 3: ADD DATASET AND TYPE DEFINITIONS TO SLR DATABASE.
/**          USE AS IS OR TAILOR WITH LOCAL VALUES.
/**
*****
**/STEP3 EXEC PGM=C9LSLR
**/STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
**/CIGPUNCH DD SYSOUT=*
**/CIGLOG DD SYSOUT=*
**/CIGIN DD *
ADD NAME RULE FOR ENDEVOR TYPE HTML CASE SENSITIVE.
ADD NAME RULE FOR ENDEVOR TYPE JAVA CASE SENSITIVE.
ADD NAME RULE FOR ENDEVOR TYPE UNIXMAKE CASE SENSITIVE .
ADD NAME RULE FOR ENDEVOR TYPE DOC CASE INSENSITIVE .
LIST NAME RULES.
/*
```

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 Endeavor.
2. Add/Delete/List a Short Name based on a given Long Name.
3. Add/Delete/List a Long Name based on a given Short Name.

The Syntax for Defining Types to the SLR

The following is the syntax used to 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 SCM Suite will monitor the transaction for the distributed object type.

ADD NAME RULE FOR ENDEVOR TYPE '*HostSCM-type*'
 case sensitive|case insensitive .

SLR Long Name Rule Syntax for Endeavor

Keyword	Description	Notes
ADD DELETE LIST NAME RULE FOR ENDEVOR TYPE <i>HostSCM-type</i>	These keywords and variable are required. The variable is a 1-8 character HostSCM-type that represents a distributed object type.	Required.
Case sensitive <u>Case insensitive</u>	This is an optional keyword that controls the representation of the distributed object name storage. Usage of this parm	Default is case insensitive. Ignored for the Delete and List verbs.

	should reflect the platform case sensitivity requirements. For instance, Unix and Linux is case sensitive, where as Windows files are not.	
--	--	--

SLR Long Name Rules Parameter Description

Example of SLR Definition Syntax:

```
ADD NAME RULE FOR ENDEVOR TYPE XLS      .
ADD NAME RULE FOR ENDEVOR TYPE UNIXMAKE CASE
SENSITIVE .
ADD NAME RULE FOR ENDEVOR TYPE DOC      CASE
INSENSITIVE .
```

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'
ENDEVOR TYPE 'Endevor-type'
.
```

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	The long name entry must exist for the DELETE and either case can be true for the LIST function. Wildcarding is not

	short name for the ADD.	allowed.
ENDEVOR TYPE 'Endevor-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.

SLR Short Name Rules Parameter Description

Example of SLR Short Name Syntax:

The following is an example of Endevor based rule definitions.

```
ADD SHORT NAME WHERE LONGNAME = 'Fiscal Year End 2000 Spread
Sheets.xls'
    Endevor type xls .
ADD SHORT NAME WHERE LONGNAME = 'OurHomePage.HTML'
    Endevor type html .
```

SLR Short Name Syntax Example

List Short Name CIGPUNCH Example:

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

```
* HEL00001
LIST SHORTNAME WHERE LONGNAME =
HELLO STEVE
```

List Long Name Output

Long Name Syntax:

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

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.

SLR Long Name Rules Parameter Description

Example of SLR Long Name Syntax:

The following is an example of Endeavor based rule definitions.

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

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.

```
* HELLO STEVE
LIST LONGNAME WHERE SHORTNAME = HEL00001 .
```

List Long Name Output

SLR in Endeavor Processors

Endeavor 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 the Cloud 9 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 Endeavor

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

Chapter 4: Exits, HTTPD Security Issues, and Customizations

There are two exit points 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
/* ----- */
```

CLZREX00 Sample Exit

CLZREXIT – C1UXSITE Support

CLZREXIT is the rexx exec for C1UXSITE, multiple C1DEFLTS switching. Please review this exit for customization. The sample exit can be found in the CGI-BIN directory of your Cloud 9 rootdir.

```
/* rexx ----- */
- Program: CLZREXIT -
- Purpose: This is the exit driver program. -
- - - - -
- Exit 1: Is ENUXSITE to be called? -
-   return '' do not call ENUXSITE -
-   return 'YES' call ENUXSITE -
-   return 'YES,DDNAME,DSNAME' call ENUXSITE and allocate -
-   ddname/dsname before call -
- - - - -
- Exit 2: Is CIGINI loader to be called? -
-   return '' do not call CIGINI loader -
-   return 'PGM' call specified program -
-   return 'PGM,DDNAME,DSNAME' call specified program and -
-   pass ddname/dsname to -
-   specified program. -
- - - - - */
----- */
parse arg xitno
select
  when (xitno == '1') then buffer = Exit01()
  when (xitno == '2') then buffer = Exit02()
end

return buffer

/* ----- */
/* *** C1DEFLTS table switching *** */
/* ENUXSITE gets called when requesting a list of environments */
/* or calling Endeavor to perform an action. */
/* ----- */
Exit01:
/* ----- */
/* Example: Do not call ENUXSITE. */
/* ----- */
buf = ''

/* ----- */
/* Example: Invoke ENUXSITE, but do not allocate a ddname. */
/* ----- */
/* buf = 'YES' */

/* ----- */
/* Example: If the user is P390Z then invoke ENUXSITE and */
/* allocate the specified ddname/dataset name prior */
/* to invoking ENUXSITE. */
/* ----- */
/* if (userid() == 'P390Z') then */
/* buf = 'YES,CIGDD01,CIGT.STEVE.LOADLIBX' */

return buf

/* ----- */
/* *** CIGINI switching *** */
/* This logic is used if the FastLIST database is used to get a */
/* list of systems, subsystems, types, or processor groups. It is */
/* also used when the FastLIST database is used to get a list of */
/* elements. */
```

```

/* ----- */
Exit02:
/* Example: Do not switch CIGINI files. */
  buf = ''

/* Example of calling exit program called CIGXSAMP */
/* buf = 'CIGXSAMP' */

/* Example of calling exit program called CIGXSAMP */
/* and have the following dd statement allocated. */
/* //CIGDD01 DD DSN=CIGT.STEVE.LOADLIB,DISP=SHR */
/* buf = 'CIGXSAMP,CIGDD01,CIGT.STEVE.LOADLIB' */

return buf

```

CLZREXIT - CIUXSITE Support

A compliment to switching C1DEFLTS would be switching CIGINI files. The CIGXSAMP program can be found in the JCLLIB offloaded during installation.

Alternate ADDTYPE Definitions Location

Due to security constraints, some customers need to store the ADDTYPE definitions in a file other than the standard HTTPD.CONF file. The ADDTYPE definitions are the only reason that Cloud 9 directly reads the HTTPD.CONF during processing. There are two options for those customers that wish to use an alternate ADDTYPE definition. The first option is in a USS directory, the second option is a OS/390 file.

The pointer to the alternate location is stored in the HTTPD.ENVVARS file and the syntax is as follows:

OS/390 file:

C9_ADDTYPE_FILE=dataset(member)

USS file:

C9_ADDTYPE_FILE=rootdir/addtypes

This parameter is included as a line in the HTTPD.ENVVARS files as follows:

```

PATH=/bin:./usr/sbin:/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/ldap/ldap
SHELL=/bin/sh
TZ=EST5EDT

```

```
LANG=C
LC_ALL=en_US.IBM-1047
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:/usr/lpp/ldap/lib/n
ls/msg
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/ldap/lib:<JA
VA_HOM
JAVA_HOME=<JAVA_HOME>
CLASSPATH=./usr/lpp/internet/server_root/CAServlet:<JAVA_HOME>/lib/classe
s.zip
STEPLIB=CURRENT
SERVER_ROOT=/u/p8887/
C9_ADDTYPE_FILE=CIGT.CONNIE.HTM(ADDTYPES)
```

ENVVARS example with alternate ADDTYPE

During a download or add to the host, the Cloud 9 will check to see if there is an C9_ADDTYPE_FILE statement included in the ENVVARS file. If there is not, then Cloud 9 reads the HTTPD.CONF file to search out the ADDTYPE statements. If there is a C9_ADDTYPE_FILE statement, then Cloud 9 attempts to access and read the file. If Cloud 9 can not find or read the file, it will make a second attempt against the ADDTYPEs in the original HTTPD.CONF file.

LSERV SUPPORT

For users with LSERV, it is necessary to tell Cloud 9 that LSERV has control over the Endeavor VSAM files. This is done through the LSERV SUBSYSTEM parameter in the CIGINI file. If LSERV is active, the user must code the LSERV SUBSYSTEM as in the example below.

```
DEFINE COMMON SECTION
PRODUCT LOADLIB      = 'FLHQ1.FLHQ2.LOADLIB'
* PRODUCT LOADLIB    = 'FLHQ1.FLHQ2.AUTHLIB'
WORK UNIT            = TDISK
VIO UNIT             = TDISK
DO NOT ALLOW ALTERNATE CIGINI FILE
ENDEAVOR CONLIB DSNAME = 'QUAL1.QUAL2.CONLIB'
JAVASERVERCONTROL DSNAME = 'FLHQ1.FLHQ2.JAVALIB'
LSERV SUBSYSTEM      = SSN$SYSA
```

LSERV Example in CIGINI

The LSERV subsystem must start with the four characters SSN\$ and the full eight character name must be equal to the expected ddname allocated in the Endeavor JCL or interactive processing. Cloud 9 will allocate the ddname for foreground actions and will include the ddname in the batch JCL.

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

Application Life Cycle Worksheet

CLZREX00 – Cloud 9
Temporary Dataset Prefix
Setting, 24
Define Breeze Approvers, 6
Email, 2, 4, 30

FTP server, 35
HTML, 30, 31
Ip-address, 35
Language is FTP1, 33
Platform Type, 35