



Guía del administrador de SCLM Developer Toolkit



Guía del administrador de SCLM Developer Toolkit

Nota

Antes de utilizar este documento, lea la información general en el apartado "Avisos de la documentación de IBM Rational Developer for System z" en la página 133.

Primera edición (septiembre de 2009)

Esta edición atañe a IBM Rational Developer para System z Versión 7.6, (número de programa 5724-T07) y a todos los releases y modificaciones posteriores, mientras no se indique lo contrario en una nueva edición.

También puede pedir publicaciones a través de su representante de IBM o de la sucursal de IBM de su localidad. En la dirección que figura más abajo no hay stock de publicaciones.

IBM agradece sus comentarios. Puede enviar comentarios por correo a:

IBM Corporation
Attn: Information Development Department 53NA
Building 501 P.O. Box 12195
Research Triangle Park NC 27709-2195
Estados Unidos de América

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo a utilizar o distribuir la información del modo que considere conveniente sin incurrir por ello en ninguna obligación para con usted.

Nota sobre los derechos restringidos de los usuarios del Gobierno de EE.UU. - El uso, la duplicación o la divulgación están sujetos a las restricciones establecidas en el contrato GSA ADP Schedule Contract con IBM Corp.

El sitio web de IBM Rational Developer para System z se encuentra en

<http://www-306.ibm.com/software/awdtools/rdz/>

La última edición de este documento siempre está disponible en este sitio web.

Copyright International Business Machines Corporation 2009. Reservados todos los derechos. Derechos restringidos para los usuarios del gobierno de EE. UU. -- Utilización, duplicación o divulgación restringidos por el GSA ADP Schedule Contract con IBM Corp.

© Copyright International Business Machines Corporation 2009.

Contenido

Tablas	v
---------------	----------

Acerca de este documento	vii
A quién va dirigido este manual	vii

Capítulo 1. Instalación del producto.	1
--	----------

Capítulo 2. Personalización de SCLM para SCLM Developer Toolkit	3
Resumen de construcciones JAVA/J2EE	3
Objetos de construcción JAVA/J2EE generados	4
Conversores de lenguaje para el soporte de JAVA/J2EE	5
Definiciones de lenguaje SCLM	5
Conjuntos de datos SCLM para JAVA/J2EE	7
Tipos SCLM	7
Atributos de conjunto de datos recomendados para algunos tipos habituales	7
Formatos de miembro SCLM	9
Formato \$GLOBAL	9
Formato ARCHDEF J2EE	10
Formato de script de construcción Ant J2EE	13
Esqueletos de construcción XML Ant JAVA/J2EE	16
Correlación de proyectos J2EE con SCLM	18
Recomendaciones para correlaciones proyectos J2EE con SCLM	18
Despliegue de SCLM Developer Toolkit	20
Despliegue de WebSphere Application Server (WAS)	21
Despliegue de SCLM en UNIX System Services	22
Despliegue seguro	22
Autenticación de clave pública	22
Otras opciones de despliegue	23
Opciones de almacenamiento ASCII o EBCDIC	23
Conversores de lenguaje ASCII/EBCDIC	23
\$GLOBAL	24
TRANSLATE.conf	26
SITE y opciones específicas de proyecto	27
Ejemplo de utilización de combinaciones de alteraciones temporales de TRANSLATE.conf	32
Ejemplo de utilización de combinaciones de alteraciones temporales de BIDIPROP	33

Capítulo 3. Soporte SQLJ.	35
¿Qué es SQLJ?	35
¿Qué es DB2?	35
¿Qué es JDBC?	35
¿Qué es SQLJ?	36
Comparación entre JDBC y SQLJ	36
¿Qué es un perfil serializado?	37
¿Qué es un DBRM?	37
Preparación del programa SQLJ	38
Conversión	38
Personalización	39

Establecimiento de enlaces	39
Tipos y conversores de SCLM DT	40
Ajuste del proceso de construcción	40
Ajuste del script de construcción	41

Capítulo 4. Seguridad de SCLM	47
Distintivo de seguridad	47
Configuración del producto de seguridad	47
Perfiles de seguridad	48
ID de usuario sustituto	48
Ejemplo: construcción	48
Ejemplo: despliegue	49

Capítulo 5. Construcciones y promociones iniciadas por CRON	51
Requisitos de STEPLIB y PATH	51
Ejecución de trabajos de construcción de CRON	52
Muestras de trabajos de construcción de CRON	52

Apéndice A. Visión general de SCLM	57
Conceptos de SCLM	57
Denominación de archivos	57
Tipo	57
Lenguaje	58
Propiedades de SCLM	58
Estructura de proyectos SCLM	58
ARCHDEF	58
Conceptos de JAVA/J2EE	59

Apéndice B. Tabla de conversión de nombres largos/cortos	61
Resumen técnico del programa de conversión de SCLM	61
Proceso de registros de nombres largos/cortos único	62
Proceso FINDLONG	62
Proceso FINDSHORT	62
Proceso TRANSLATE	63
Proceso de registros de nombres largos/cortos múltiple	63
Proceso IMPORT	64
Proceso MIGRATE	64

Apéndice C. API de SCLM Developer Toolkit	65
Formato de invocación	65
Esquema XML para mandatos SCLMDT	66
Funciones y parámetros de solicitud	68
Formato de función	68
Lista de funciones	68
AUTHUPD – Cambiar código de autoridad SCLM	69
BROWSE – Examinar miembro SCLM	70
BUILD – Miembro SCLM de construcción	70
DELETE – Suprimir un miembro SCLM	71

DEPLOY – Desplegar un archivo EAR J2EE	72
EDIT – Editar miembro SCLM	73
INFO – Información sobre el estado del miembro SCLM	73
J2EEIMP – Importar proyecto desde SCLM	74
J2EEMIG – Migrar proyecto en SCLM	75
J2EEMIGB – Migrar el proyecto por lotes en SCLM	76
JARCOPY – Copiar archivo JAR	77
JOBSTAT – Recuperar estado de trabajo por lotes	78
LRECL – Recuperar LRECL de conjunto de datos SCLM	78
MIGDSN – Listar conjuntos de datos y miembros que no son de SCLM	78
MIGPDS – Migrar conjuntos de datos y miembros que no son de SCLM en SCLM	79
PROJGRPS – Recuperar grupos SCLM para un proyecto	79
PROJINFO – Recuperar información de proyecto SCLM	79
PROMOTE – Promover un miembro SCLM.	80
REPORT – Crear informe de proyecto	81
REPUTIL – Informe DBUTIL de SCLM	82
SAVE – Guardar miembro SCLM	82
UNLOCK – Desbloquear miembro SCLM	83
UPDATE – Actualizar información sobre el miembro SCLM	83
VERBROW – Versiones de navegación de SCLM	84
VERDEL – Versiones de supresión de SCLM	84
VERHIST – Historial de versiones de SCLM	85
VERLIST – Versiones de lista de SCLM	85
VERREC – Versiones de recuperación de SCLM	86
Muestras	86
sclmdt_request.xml	86
xmlbld.java	87
sclmdt_response.xml	89

Apéndice D. Programa de utilidad de construcción de Rational Application Developer for WebSphere Software . . . 97

Visión general del Programa de utilidad de construcción de Rational Application Developer for WebSphere Software	97
Almacenamiento de objetos Java/J2EE en SCLM	98
El Programa de utilidad de construcción comparado con el proceso de construcción ANT nativo	98
Notas de instalación del Programa de utilidad de construcción de Rational Application Developer for WebSphere Software	99
Integración de SCLM con el Programa de utilidad de construcción	99
Implementación y utilización del Programa de utilidad de construcción de Rational Application Developer for WebSphere Software	99
Conversores de lenguaje de Programa de utilidad de construcción de SCLM	100
Scripts y formatos de construcción del Programa de utilidad de construcción.	103
Soporte para la construcción SQLJ	112
Fuente Java en archivos de archivado	116
CASO DE EJEMPLO DE UTILIZACIÓN: 'PlantsByWebSphere'	116

Apéndice E. BUILD FORGE y SCLM 123

Visión general	123
Prerrequisitos	123
Cómo invocar el agente Build Forge en z/OS	123
Configuración del servidor de consola Build Forge	124
Ejemplo de promoción SCLM	130

Avisos de la documentación de IBM Rational Developer for System z . . . 133

Licencia de copyright.	134
Reconocimientos de marcas registradas.	135

Índice. 137

Tablas

1.	Lista de comprobación del administrador de SCLM.	1
2.	Conversores de SCLM Developer Toolkit . . .	5
3.	Variables definidas por el cliente	13
4.	Conversores de lenguaje SCLM y ASCII/EBCDIC	23
5.	Variables \$GLOBAL.	25
6.	Opciones SITE/Project	30
7.	Comparación entre JDBC y SQLJ	36
8.	Tipos de conversores SCLM para SQLJ	40
9.	Propiedades de sqlj.*	41
10.	propiedades de db2sqljcustomize.*.	42
11.	Perfiles de seguridad de SCLMDT Developer Toolkit	48
12.	Parámetros de conversión de nombres largos/cortos.	62
13.	Parámetros projectImport	104
14.	Parámetros projectBuild	104
15.	Parámetros EJBexport.	105
16.	Parámetros WARExport	106
17.	Parámetros EARExport	106
18.	Parámetros AppClientExport	106

Acerca de este documento

Este documento aborda la configuración de IBM® Software Configuration and Library Manager (SCLM) necesaria para la función SCLM Developer Toolkit de IBM Rational Developer for System z.

En adelante, en este manual se utilizarán los siguientes nombres:

- *IBM Software Configuration and Library Manager* se denominará *SCLM*.
- *IBM Rational Developer para System z* se denominará *Developer para System z*.
- La función *SCLM Developer Toolkit* de *IBM Rational Developer para System z* se denominará *SCLM Developer Toolkit* y, en ocasiones, se abreviará como *Developer Toolkit* o *SCLMDT*.

A quién va dirigido este manual

Este documento contiene información para el administrador acerca de los proyectos SCLM que se utilizarán con SCLM Developer Toolkit. Dicha información incluye proyectos que utilizan lenguajes de componentes de Java™ y z/OS UNIX® System Services, además de proyectos SCLM tradicionales.

Estos administradores deberían estar familiarizados con z/OS UNIX System Services, scripts de REXX y el compilador de Java, así como con proyectos SCLM y definiciones de lenguaje.

Capítulo 1. Instalación del producto

Esta publicación no aborda la implementación y la carga del producto SCLM, que se suministra con el sistema operativo z/OS. Tampoco aborda la instalación y la configuración del propio SCLM Developer Toolkit, que es una función de Rational Developer for System z.

Consulte los manuales *ISPF Software Configuration and Library Manager Project Manager's and Developer's Guide* (SC34-4817) y *Rational Developer para System z Guía de configuración de host* (SC11-3613) para obtener más información sobre estas tareas.

Para concluir las tareas de personalización y de definición del proyecto, el administrador de SCLM debe conocer de antemano varios valores personalizables de Developer for System z, tal como se describe en la Tabla 1. Póngase en contacto con el programador del sistema z/OS responsable de la instalación y de la personalización de Developer for System z para obtener más información.

Tabla 1. Lista de comprobación del administrador de SCLM

Descripción	Valor predeterminado	Dónde buscar la respuesta	Valor
Biblioteca de muestra de Developer for System z	FEK.SFEKSAMP	Instalación SMP/E	
Directorio de muestra de Developer for System z	/usr/lpp/rdz/samples	Instalación SMP/E	
Directorio bin de Java	/usr/lpp/java/J5.0/bin	rsed.envvars - \$JAVA_HOME/bin	
Directorio bin de Ant	/usr/lpp/Ant/apache-ant-1.7.1/bin	rsed.envvars - \$ANT_HOME/bin	
Directorio padre WORKAREA	/var/rdz	rsed.envvars - \$_CMDSERV_CONF_HOME	
Directorio padre de configuración de proyecto SCLMDT	/etc/rdz/sclmdt	rsed.envvars	
Conversión de nombre largo/corto VSAM	FEK.#CUST.LSTRANS.FILE	rsed.envvars - \$_SCLMDT_TRANTABLE	

Capítulo 2. Personalización de SCLM para SCLM Developer Toolkit

Este capítulo estudia la forma en que el administrador de SCLM puede personalizar SCLM para que lo utilice SCLM Developer Toolkit.

Resumen de construcciones JAVA/J2EE

Aquí obtendrá un resumen del proceso que tiene lugar para las construcciones Java y J2EE utilizando los conversores suministrados.

Nota: Puede construir directamente miembros JAVA/J2EE o ARCHDEFS en TSO/ISPF en el host, además de mediante el cliente Developer Toolkit.

La ARCHDEF contiene los miembros que constituyen el proyecto JAVA/J2EE y son una representación con nombre corto del modo en que el proyecto existe en un espacio de trabajo de Eclipse.

Se construye la propia ARCHDEF, que invoca a un conversor de lenguaje de verificación previo a la construcción (J2EEANT). El conversor lee el script de construcción J2EE, referenciado en la ARCHDEF mediante la palabra clave SINC, y superpone las propiedades especificadas en el XML Ant de esqueleto referenciado por las propiedades SCLM-ANTXML (A). El script de construcción, cuando lo genera SCLM Developer Toolkit, se almacena en SCLM con un lenguaje de J2EEANT (1).

Una ARCHDEF genera clases Java para la fuente Java identificada con la palabra clave INCLD en la ARCHDEF (2) y en cada ARCHDEF también puede generar un archivo de archivado J2EE, como por ejemplo un archivo JAR, WAR o EAR. El objeto J2EE creado depende del script de construcción apropiado referenciado y utiliza la palabra clave OUT1 de ARCHDEF (3), (B).

Cuando se construye la ARCHDEF, se ejecuta el conversor de lenguaje de verificación previo a la construcción asociado con el script de construcción (en SCLM tipo J2EEBLD) y determina qué partes de la ARCHDEF se deben reconstruir (incluidas las ARCHDEF anidadas (4) identificados mediante el uso de la palabra clave ICL en la ARCHDEF). A continuación se copian dichas partes en el área de trabajo del sistema de archivos de z/OS UNIX System Services y Ant compila y genera los objetos JAVA/J2EE necesarios especificados por el script de construcción y ARCHDEF. Cualquier referencia jar o class (clase) externa que el proyecto IDE deba resolver se lleva a cabo desde la vía de acceso definida en la propiedad CLASSPATH_JARS (C).

A continuación, SCLM procesa cada componente ARCHDEF individual que ejecute cada conversor de lenguaje asociado con dicho componente. El conversor de lenguaje JAVA, asociado con la fuente Java, copia de nuevo los archivos de clases (class) creados en SCLM.

Finalmente, el conversor ARCHDEF determina qué objetos J2EE se han generado (JAR, WAR, EAR) y vuelve a copiar estas partes en SCLM.

Resulta esencial crear una ARCHDEF independiente para cada componente de la aplicación que pueda constituir una aplicación empresarial (EAR). Es decir, un EAR que contenga un WAR que contenga un JAR EJB debería tener una ARCHDEF para el JAR, una ARCHDEF para el WAR con un INCL de la ARCHDEF de JAR EJB. A continuación, la ARCHDEF de EAR debería incluir un INCL de la ARCHDEF de WAR.

El ejemplo siguiente muestra el código JAR:

```
*
* Inicialmente generado el 10/05/2006 por SCLM DT V2
*
LKED J2EE0BJ * Conversor de construcción J2EE
*
* Fuente que debe incluirse en la construcción
*
INCLD AN000002 V2TEST * com/Angelina.java *
INCLD V2000002 V2TEST * com/V2Java1.java (2) *
INCLD V2000003 V2TEST * V2InnerClass.java *
*
* JAR controlados por SCLM anidados que deben incluirse *
*
INCL V2JART1 ARCHDEF * DateService.jar (4) *
*
* Script de construcción y salidas generadas
*
SINC V2JARB1(1) J2EEBLD * Script de construcción JAR J2EE *
OUT1 * J2EEJAR * V2TEST.jar (3) *
LIST * J2EELIST
```

Figura 1. ARCHDEF de aplicación Jar (JAR) de muestra

El ejemplo siguiente muestra el script JAR correspondiente.

```
<ANTXML>
<project name="JAVA Project" default="jar" basedir=".">
<property name="env" environment="env" value="env"/>
<property name="SCLM_ARCHDEF" value="V2JAR1"/>
<property name="SCLM_ANTXML" value="BWBJAVA" /> (A)
<property name="SCLM_BLDMAP" value="YES"/>
<property name="JAR_FILE_NAME" value="V2TEST.jar"/> (B)
<property name="CLASSPATH_JARS" value="/var/SCLMDT/CLASSPATH"/> (C)
<property name="ENCODING" value="IBM-1047"/>
</ANTXML>
```

Figura 2. Muestra JAR de script de construcción J2EE

Objetos de construcción JAVA/J2EE generados

Se generan los siguientes objetos:

- Compilación de todas las fuentes de Java en clases de salida, almacenadas en el tipo SCLM JAVACLAS.
- Clases almacenadas en SCLM y nombre largo/corto almacenado en tablas convertibles.
- Jar opcional creado (contiene clases y es posible que contenga otros componentes de proyecto Java tales como XML, HTML, etc. en una estructura empaquetada).
- Objetos Jar almacenados en SCLM y nombre largo/corto almacenado en tabla convertible.
- Estructura Jar determinada por la ARCHDEF utilizada. Los nombres largos asociados con los miembros de la ARCHDEF determinan el formato de empaquetado de Jar.

- JAR EJB opcional (contiene clases y es posible que contenga otros componentes de proyecto Java tales como XML, HTML, JSP, etc. en una estructura empaquetada).
- Archivo WAR web opcional basado en el archivo J2EE web.xml del proyecto J2EE y almacenado en SCLM tal como se explica más arriba.
- Archivo EAR opcional para el despliegue basado en application.xml en el proyecto J2EE y almacenado en SCLM tal como se explica más arriba.
- Todas las salidas de listados se almacenan en el tipo SCLM J2EELIST.

Conversores de lenguaje para el soporte de JAVA/J2EE

SCLM Developer Toolkit necesita nuevos conversores de lenguaje nuevos definidos en SCLM para el soporte de JAVA/J2EE. Estos conversores de lenguaje se envían en los miembros de FEK.SFEKSAMV tal como se muestra a continuación:

Tabla 2. Conversores de SCLM Developer Toolkit

Conversor	Descripción
BWBTRANJ	Conversor de miembro predeterminado de muestra. Ningún análisis. Parecido a SCLM FLM@TEXT. Este conversor se puede personalizar para crear las definiciones de lenguaje J2EEPART, J2EEBIN, BINARY y TEXT.
BWBTRANS	Conversor de lenguaje SQLJ de muestra. LANG=SQLJ
BWBTRAN1	Conversor de lenguaje Java de muestra. LANG=JAVA
BWBTRAN2	Conversor de lenguaje JAVA/J2EE de muestra que incorpora Ant (para varios compiladores Java y compilaciones JAR, WAR y EAR)
BWBTRAN3	Conversor de lenguaje J2EE de muestra para el soporte de J2EE de ARCHDEF de SCLM. LANG=J2EEOBJ

El administrador de SCLM deberá copiar estas muestras, renombrarlas si es necesario y, a continuación, generarlas en la biblioteca PROJDEFS.LOAD para cada proyecto SCLM en el que se necesite el soporte de Java. Estos conversores deben añadirse o compilarse en la Definición del proyecto.

Se suministra una definición de proyecto de ejemplo para proyectos JAVA/J2EE y componentes de host en la muestra BWBSCLM.

Definiciones de lenguaje SCLM

Los conversores de muestra definen los siguientes lenguajes:

J2EEANT

Conversor de construcción principal para construcciones JAVA/J2EE; este conversor de verificación se invoca cuando se construye una ARCHDEF J2EE. Se invoca al conversor porque el script de construcción JAVA/J2EE, almacenado en el tipo SCLM J2EEBLD, se guarda en SCLM con un lenguaje de J2EEANT. A continuación se le referencia mediante la palabra clave SINC en la ARCHDEF.

Este conversor de verificación determina qué partes deben construirse (incluidas las ARCHDEF anidadas) y, en función de las modalidades de construcción, copia dichas partes en el directorio WORKAREA de z/OS UNIX System Services. Se personaliza automáticamente un XML Ant de esqueleto de acuerdo con el script de construcción y las partes construidas en el área de trabajo utilizando Ant. Estos archivos de clases (class) se pasan a los conversores de lenguaje JAVA/JAVABIN para volver a

almacenar los archivos de clases en SCLM. Los objetos J2EE generados tales como un JAR, WAR o EAR se pasan al conversor de lenguaje ARCHDEF (J2EEOBJ) para que se vuelvan a almacenar en SCLM.

J2EEBIN

Tipo de lenguaje que especifica un componente almacenado JAVA/J2EE binario o ASCII y está definido por BWBTRANJ de muestra. No se produce ningún análisis en particular al construir esta definición de lenguaje. Los archivos binarios JAVA/J2EE y los archivos de texto que desee que se almacenen como ASCII pueden insertarse genéricamente bajo esta definición de lenguaje si no se necesita ningún análisis de construcción en particular.

J2EEOBJ

Conversor de construcción final que se invoca como parte del proceso de construcción de ARCHDEF. Este conversor determina qué objetos J2EE (JAR, WAR, EAR) se construyeron previamente en el conversor J2EEANT y los copia en SCLM con el nombre corto generado que se ha suministrado. Este conversor se referencia mediante la palabra clave LKED en la propia ARCHDEF.

J2EEPART

Tipo de lenguaje que especifica un componente JAVA/J2EE y está definido por BWBTRANJ de muestra. No se produce ningún análisis en particular al construir esta definición de lenguaje. Los componentes que no son de fuente Java ni J2EE que necesitan conversión de lenguaje ASCII/EBCDIC pueden insertarse genéricamente bajo esta definición de lenguaje si no se necesita ningún análisis de construcción en particular (por ejemplo, html, XML, .classpath, .project o tablas de definición). Se puede utilizar una definición de lenguaje opcional de TEXT.

JAVA Tipo de lenguaje para la fuente Java y definido por BWBTRAN1 de muestra. El conversor de Java determina qué tipo de construcción se ha emitido contra la fuente Java.

Nota: Esta definición de lenguaje debe asignarse a programas Java si desea almacenar la fuente Java en EBCDIC en el host (es decir, es posible visualizar y editar directamente la fuente en el host a través de ISPF). La ventaja de definir programas con esta definición de lenguaje es la posibilidad de poder editar y visualizar la fuente directamente en el host de z/OS. Las desventajas son que las conversiones de página de códigos deben realizarse al migrar o importar proyectos desde el cliente al host.

- Caso de ejemplo 1: Construcción emitida contra programa Java individual.

El conversor de Java compila la fuente en clases de salida. La clase se almacena en SCLM en el tipo JAVACLAS. La salida de compilación de Java se almacena en el tipo JAVALIST.

Cualquier dependencia de la vía de acceso de clases se puede satisfacer almacenando JAR dependientes en el directorio de la vía de acceso de clases especificado en el parámetro de miembro \$GLOBAL CLASSPATH_JARS. Para obtener más información, consulte “\$GLOBAL” en la página 24.

- Escenario 2: La construcción en ARCHDEF (ARCHDEF llama al script de construcción J2EEANT referenciado por la palabra clave SINC) deja que el script de Ant especificado realice la construcción. El propio conversor de Java, cuando lo invoca

ARCHDEF, sólo copia las clases de salida en SCLM. Se almacena un archivo de resumen de construcción ANT en JAVALIST. Los componentes Java individuales tienen una tabla de salida almacenada en JAVALIST.

JAVABIN

Tipo de lenguaje parecido a Java que se utiliza al almacenar la fuente Java como ASCII en SCLM.

SQLJ Tipo de lenguaje para el código fuente SQLJ definido por BWBTRANS de muestra. Los Miembros de SQLJ definidos con este conversor de lenguaje invocan al conversor de lenguaje SQLJ durante el tiempo de construcción. La fuente SQLJ se convierte en fuente Java y se compila en clases y objetos serializados (archivos .ser) en el tipo SQLJSER. Opcionalmente, los miembros de DBRM también se pueden generar en el tipo DBRMLIB.

Nota: Todos los objetos como JAR, WAR y EAR tienen sus partes fuente comprimidas internas en ASCII para distribuir las en todas las plataformas.

Conjuntos de datos SCLM para JAVA/J2EE

Es recomendable que cree conjuntos de datos de origen/destino de SCLM de RECFM=VB, LRECL=1024 para cualquier fuente JAVA/J2EE que deba almacenarse en SCLM desde el cliente Toolkit para permitir tipos de registro largos.

Los editores del cliente basado en Eclipse crean archivos de longitud de registro variable y, para mantener la integridad, los conjuntos de datos de destino de host en SCLM también deberían ser de RECFM=VB. La utilización de conjuntos de datos de longitud de registro fija (RECFM=FB) tendrá como consecuencia que los miembros importados tengan espacios en blanco anexados al final del registro.

Tipos SCLM

Existe una serie de tipos SCLM que deben crearse para el soporte de JAVA/J2EE. Algunos de estos tipos son tipos obligatorios y deben crearse para que el soporte de JAVA/J2EE funcione.

Atributos de conjunto de datos recomendados para algunos tipos habituales

Se recomiendan los atributos de conjunto de datos de DSORG=PO TRACKS(1,5) DIR=50 BLKSIZE=0 (determinados por el sistema) para los siguientes Tipos SCLM.

Además, se recomiendan los siguientes atributos de formato de registro y de longitud de registro:

Tipo SCLM	RECFM	LRECL
ARCHDEF	FB	80
J2EEBLD	FB	256
JAVALIST	VB	255
J2EELIST	VB	255
DBRMLIB	VB	256
JAVACLAS	VB	256
J2EEEAR	VB	256

Tipo SCLM	RECFM	LRECL
J2EEJAR	VB	256
J2EEWAR	VB	256
SQLJSER	VB	256
Tipos de conjunto de datos fuente adicionales para Java/J2EE	VB	1024

ARCHDEF

El tipo ARCHDEF contiene miembros ARCHDEF JAVA/J2EE.

Las partes de nombre largo de cada miembro ARCHDEF esquematizan la estructura de proyectos JAVA/J2EE. La ARCHDEF para un proyecto determinado se puede crear dinámicamente desde el cliente al migrar en proyectos nuevos o se puede actualizar al añadir partes nuevas a un proyecto existente.

La ARCHDEF de SCLM es el archivo SCLM primario para definir los elementos de un proyecto JAVA/J2EE. Con respecto a las aplicaciones JAVA/J2EE, la ARCHDEF representa cómo se estructura la aplicación J2EE en el espacio de trabajo de proyecto IDE de cliente.

La estructura de archivo de proyecto de la aplicación se replica en la ARCHDEF (utilizando el nombre corto del host SCLM para correlacionar la estructura de nombre largo). Las palabras clave adicionales de la ARCHDEF tales como LINK, SINC y OUT1 indican a SCLM la naturaleza J2EE de este proyecto y la fuente incluye un script de construcción JAVA/J2EE para facilitar el proceso de construcción de dicho proyecto.

J2EEBLD

El tipo J2EEBLD es necesario para los procesos de construcción y despliegue de Java y J2EE y contiene lo siguiente:

- Scripts de construcción J2EEBLD utilizados para dirigir el proceso de construcción y despliegue de Ant.
- Scripts Java y ANTXML J2EE que deben invocarse para construcciones y despliegues.
- \$GLOBAL, que especifica las propiedades predeterminadas para el proyecto SCLM para el proceso de construcción JAVA/J2EE. Consulte "Formato \$GLOBAL" en la página 9 y "\$GLOBAL" en la página 24 para obtener más información.

Nota: Se suministran scripts Java y ANTXML J2EE de muestra.

Generalmente, estos scripts no necesitan ser personalizados o requieren muy poca personalización. Las variables dependientes de usuario y sitio se personalizan en los propios scripts J2EEBLD para alterar temporalmente las variables ANTXML predeterminadas. Para obtener más información, consulte "Esqueletos de construcción XML Ant JAVA/J2EE" en la página 16.

JAVALLIST

El tipo JAVALLIST es necesario para el proceso de construcción Java y contiene salidas de listados de construcciones Java.

J2EELIST

El tipo J2EELIST es necesario para el proceso de construcción J2EE y contiene salidas de listados de construcciones J2EE.

DBRMLIB

Técnicamente, es un tipo DB2.

DBRMLIB es necesario para el soporte SQLJ y almacena los módulos de petición de base de datos.

JAVACLAS

El tipo JAVACLAS es necesario para los procesos de construcción Java y J2EE y contiene archivos de clases (class) de salida de construcciones asociadas con definiciones de lenguaje JAVA y J2EEANT.

J2EEEAR

El tipo J2EEEAR es necesario para el proceso de construcción J2EE y contiene una salida EAR de construcciones asociadas con la definición de lenguaje J2EEANT.

J2EEJAR

El tipo J2EEJAR es necesario para construcciones JAVA/J2EE y contiene una salida JAR de construcciones asociadas con la definición de lenguaje J2EEANT.

J2EEWAR

El tipo J2EEWAR es necesario para el proceso de construcción J2EE y contiene una salida WAR de construcciones asociadas con la definición de lenguaje J2EEANT.

SQLJSER

El tipo SQLJSER es necesario para el proceso de construcción J2EE/SQLJ y almacena perfiles serializados SQLJ.

<Tipos Java/J2EE>

Se necesita un tipo SCLM independiente para que cada proyecto JAVA/J2EE se almacene en SCLM. Está previsto para evitar conflictos en archivos con el mismo nombre que se pueden producir en proyectos JAVA/J2EE. Para obtener más información, consulte “Correlación de proyectos J2EE con SCLM” en la página 18.

Formatos de miembro SCLM

Esta sección describe los formatos de miembro SCLM.

Formato \$GLOBAL

El formato \$GLOBAL es del tipo J2EEBLD y lenguaje J2EEPART. Debe utilizar el nombre \$GLOBAL y las variables se definen en formato de lenguaje con códigos.

\$GLOBAL especifica las propiedades predeterminadas para el proyecto SCLM para el proceso de construcción JAVA/J2EE. Debe almacenarse en el tipo SCLM J2EEBLD.

Para obtener una descripción de los parámetros de miembro \$GLOBAL, consulte “\$GLOBAL” en la página 24.

Consulte el ejemplo de código siguiente:

```

<property name="ANT_BIN" value="/usr/lpp/Ant/apache-Ant-1.6.0/bin/Ant"/>
<property name="JAVA_BIN" value="/usr/lpp/java/IBM/J1.4/bin"/>
<property name="_SCLMDT_CONF_HOME" value="/etc/rdz/sclmdt/CONFIG"/>
<property name="_SCLMDT_WORK_HOME" value="/var/rdz/WORKAREA"/>
<property name="CLASSPATH_JARS" value="/var/rdz/CLASSPATH"/>

```

Figura 3. \$GLOBAL - Variables de entorno SCLMDT

Formato ARCHDEF J2EE

El formato ARCHDEF J2EE es del tipo ARCHDEF y lenguaje ARCHDEF.

La ARCHDEF utiliza palabras clave de arquitectura SCLM estándar para decir a SCLM cómo debe procesar la construcción del ARCHDEF.

```

LKED J2EEOBJ
INCLD ArchivoFuente TipoFuente
INCL NombreArchdef TipoArchdef
SINC NombreScriptConstrucción J2EEBLD
OUT1 * TipoObjetoSalidaJ2EE
LIST * J2EELIST

```

LKED

Indica que se trata de una ARCHDEF LEC y proporciona el lenguaje del conversor de la ARCHDEF que debe invocarse (para las ARCHDEF J2EE siempre es J2EEOBJ).

INCLD

Inclusión SCLM del componente J2EE. *ArchivoFuente* es el nombre del miembro fuente (por ejemplo, fuente Java) que se incluye en esta ARCHDEF. *TipoFuente* es el tipo SCLM que contiene el miembro. En una ARCHDEF generada mediante SCLM Developer Toolkit encontrará un comentario indicando el nombre de archivo completo del archivo tal como existía en el proyecto en el entorno de trabajo.

INCL Inclusión SCLM de otra ARCHDEF anidada, como la ARCHDEF que contiene el manifiesto para una aplicación EJB.

SINC Inclusión de fuente del script de construcción J2EEBLD. *NombreScriptConstrucción* es el nombre del script de construcción. El tipo de fuente siempre es J2EEBLD.

OUT1

Indica el tipo de objeto J2EE creado por esta ARCHDEF. El nombre del miembro siempre es *. El *TipoObjetoSalidaJ2EE* se establece en J2EEEAR, J2EEWAR o J2EEJAR. Al miembro creado se le otorgará un nombre sacado del nombre corto generado para el archivo JAR, WAR o EAR.

LIST Listado de componentes de resumen y auditoría de la ARCHDEF construida. El nombre del miembro siempre es *. El tipo de fuente siempre es J2EELIST. Al miembro creado se le otorgará un nombre con el mismo valor que el nombre del miembro de la ARCHDEF.

Ejemplos de ARCHDEF J2EE

El ejemplo siguiente muestra el código JAR:

```

*
* Inicialmente generado el 10/05/2006 por SCLM DT V2
*
LKED  J2EE0BJ          * Conversor de construcción J2EE
*
* Fuente que debe incluirse en la construcción
*
INCLD AN000002 V2TEST  * com/Angelina.java          *
INCLD V2000002 V2TEST  * com/V2Java1.java          *
INCLD V2000003 V2TEST  * V2InnerClass.java          *
*
* Script de construcción y salidas generadas
*
SINC  V2JARB1 J2EEBLD  * Script de construcción JAR J2EE  *
OUT1  *        J2EEJAR  * V2TEST.jar                  *
LIST  *        J2EELIST

```

Figura 4. ARCHDEF de aplicación Jar (JAR) de muestra

El ejemplo siguiente muestra el código WAR:

```

*
* Inicialmente generado el 5 de septiembre de 2006 por SCLM DT V2
*
LKED  J2EE0BJ          * Conversor de construcción J2EE
*
* Fuente que debe incluirse en la construcción
*
INCLD DA000026 SAMPLE5 * JavaSource/service/dateController.java *
INCLD XX000001 SAMPLE5 * .classpath                      *
INCLD XX000002 SAMPLE5 * .project                      *
INCLD XX000003 SAMPLE5 * .websettings                    *
INCLD XX000004 SAMPLE5 * .website-config                 *
INCLD OP000002 SAMPLE5 * WebContent/operations.html      *
INCLD MA000001 SAMPLE5 * WebContent/META-INF/MANIFEST.MF *
INCLD IB000001 SAMPLE5 * WebContent/WEB-INF/ibm-web-bnd.xmi *
INCLD IB000002 SAMPLE5 * WebContent/WEB-INF/ibm-web-ext.xmi *
INCLD WE000001 SAMPLE5 * WebContent/WEB-INF/web.xml      *
INCLD MA000002 SAMPLE5 * WebContent/theme/Master.css     *
INCLD BL000001 SAMPLE5 * WebContent/theme/blue.css       *
INCLD BL000002 SAMPLE5 * WebContent/theme/blue.html      *
INCLD LO000013 SAMPLE5 * WebContent/theme/logo_blue.gif *
*
* Script de construcción y salidas generadas
*
SINC  SAMPLE5 J2EEBLD  * Script de construcción WAR J2EE  *
OUT1  *        J2EEWAR  * Sample5.war                    *
LIST  *        J2EELIST

```

Figura 5. ARCHDEF de aplicación web (WAR) de muestra

El ejemplo siguiente muestra el código EJB:

```

LKED  J2EE0BJ
*
INCLD XX000001 SAMPLE3 * .classpath *
INCLD XX000002 SAMPLE3 * .project *
INCLD MA000004 SAMPLE3 * ejbModule/META-INF/MANIFEST.MF *
INCLD EJ000004 SAMPLE3 * ejbModule/META-INF/ejb-jar.xml *
INCLD IB000003 SAMPLE3 * ejbModule/META-INF/ibm-ejb-jar-bnd.xmi *
INCLD XX000008 SAMPLE3 * ejbModule/com/ibm/ejs/container/_EJSWrapper *
* _Stub.java *
INCLD XX000009 SAMPLE3 * ejbModule/com/ibm/ejs/container/_EJSWrapper *
* _Tie.java *
INCLD XX000010 SAMPLE3 * ejbModule/com/ibm/websphere/csi/_CSIServant *
* _Stub.java *
INCLD XX000011 SAMPLE3 * ejbModule/com/ibm/websphere/csi/_Transactio *
* nalObject_Stub.java *
INCLD DA000005 SAMPLE3 * ejbModule/myEJB/DateBean.java *
INCLD DA000006 SAMPLE3 * ejbModule/myEJB/DateBeanBean.java *
INCLD DA000007 SAMPLE3 * ejbModule/myEJB/DateBeanHome.java *
INCLD EJ000001 SAMPLE3 * ejbModule/myEJB/EJSRemoteStatelessDateBeanH *
* ome_1a4c4c85.java *
INCLD EJ000002 SAMPLE3 * ejbModule/myEJB/EJSRemoteStatelessDateBean_ *
* _1a4c4c85.java *
INCLD EJ000003 SAMPLE3 * ejbModule/myEJB/EJSStatelessDateBeanHomeBea *
* nHomeBean_1a4c4c85.java *
INCLD XX000012 SAMPLE3 * ejbModule/myEJB/_DateBeanHome_Stub.java *
INCLD XX000013 SAMPLE3 * ejbModule/myEJB/_DateBean_Stub.java *
INCLD XX000014 SAMPLE3 * ejbModule/myEJB/_EJSRemoteStatelessDateBean *
* Home_1a4c4c85_Tie.java *
INCLD XX000015 SAMPLE3 * ejbModule/myEJB/_EJSRemoteStatelessDateBean *
* _1a4c4c85_Tie.java *
INCLD XX000016 SAMPLE3 * ejbModule/org/omg/stub/javax/ejb/_EJBHome_S *
* ub.java *
INCLD XX000017 SAMPLE3 * ejbModule/org/omg/stub/javax/ejb/_EJBObject *
* _Stub.java *
INCLD XX000018 SAMPLE3 * ejbModule/org/omg/stub/javax/ejb/_Handle_St *
* ub.java *
INCLD XX000019 SAMPLE3 * ejbModule/org/omg/stub/javax/ejb/_HomeHandl *
* e_Stub.java *
INCLD DA000008 SAMPLE3 * ejbModule/services/DateBeanServices.java *
INCLD XX000020 SAMPLE3 * ejbModule/services/_DateBeanServices_Stub.j *
* ava *
*
SINC SAMPLE3 J2EEBLD * Script de construcción de JAR EJB J2EE *
OUT1 * J2EEJAR * DateService.jar *
*
LIST * J2EELIST

```

Figura 6. ARCHDEF de aplicación EJB (EJB) de muestra

El ejemplo siguiente muestra el código EAR:

```

LKED  J2EE0BJ
*
INCLD XX000001 SAMPLE6 * .classpath *
INCLD XX000002 SAMPLE6 * .project *
INCLD AP000001 SAMPLE6 * META-INF/application.xml *
INCL SAMPLE3 ARCHDEF * DateService.jar *
INCL SAMPLE5 ARCHDEF * Sample5.war *
*
SINC SAMPLE6 J2EEBLD * J2EE EAR Build script *
OUT1 * J2EEEAR * Sample6.ear *
LIST * J2EELIST

```

Figura 7. ARCHDEF de aplicación EAR (EAR) de muestra

Formato de script de construcción Ant J2EE

El formato de script de construcción Ant J2EE es del tipo J2EEBLD y lenguaje J2EEANT. Puede tener cualquier nombre de hasta ocho caracteres y las variables se definen en formato de lenguaje con códigos. Los scripts de construcción son muy parecidos para JAR, WAR y EAR. La sintaxis que aparece a continuación representa un script de construcción WAR. Para JAR y EAR, las variables de scripts de construcción son las mismas excepto por la utilización de EAR_NAME y JAR_NAME en lugar de WAR_NAME.

Consulte el ejemplo siguiente que muestra un script de construcción:

```
<ANTXML>
<project name="J2EE Project type" default="web-war" basedir=".">
  <property name="env" environment="env" value="env"/>
  <property name="SCLM_ARCHDEF" value="ARCHDEF name"/>
  <property name="SCLM_ANTXML" value="ANTXML name"/>
  <property name="SCLM_BLDMAP" value="Include Buildmap"/>
  <property name="JAVA_SOURCE" value="Include Java Source"/>
  <property name="J2EE_HOME" value="{env.J2EE_HOME}"/>
  <property name="JAVA_HOME" value="{env.JAVA_HOME}"/>
  <property name="CLASSPATH_JARS" value="Classpath Directory location"/>
  <property name="CLASSPATH_JARS_FILES" value="Jar/class filenames"/>
  <property name="ENCODING" value="Codepage"/>
  <property name="DEBUG_MODE" value="debug_mode"/>

  <!-- Nombre de archivo WAR que debe crear este proceso de construcción -->
  <!-- incluir sufijo de .war -->
  <property name="WAR_NAME" value="War name" />

  <path id="build.class.path">
    <pathelement location="."/>
    <pathelement location="{J2EE_HOME}/lib/j2ee.jar"/>
    <pathelement location="{CLASSPATH_JARS}/jdom.jar"/>
    <fileset dir="." includes="**/*.jar"/>
    <fileset dir="{CLASSPATH_JARS}" includes="**/*.jar, **/*.zip"/>
  </path>

</ANTXML>
```

Figura 8. Script de construcción Ant J2EE

Los scripts de construcción SCLM superponen dinámicamente variables definidas por el usuario cuando lo pide la construcción al ejecutar el script de construcción Ant. Estas variables se establecen en los valores mostrados en la Tabla 3.

Tabla 3. Variables definidas por el cliente

Variable	Descripción
Nombre de proyecto J2EE	Tipo de proyecto Java/J2EE que se está construyendo. Se trata de un nombre de proyecto temporal establecido en el script de construcción para que Ant lo utilice durante la construcción. Se establecerá en los siguientes valores: <ul style="list-style-type: none">• Proyecto EAR J2EE• Proyecto WAR J2EE• Proyecto EJB J2EE• Proyecto JAVA No es necesario personalizar esta variable.
SCLM_ARCHDEF	Nombre de la ARCHDEF o la ARCHDEF que se está construyendo
SCLM_ANTXML	Nombre de XML Ant de esqueleto que debe utilizarse para la construcción

Tabla 3. Variables definidas por el cliente (continuación)

Variable	Descripción
SCLM_BLDMAP	Valor de Sí o No. Si es Sí, incluya la correlación de construcción SCLM en el directorio MANIFEST en JAR, WAR o EAR. Proporciona una correlación de auditoría y construcción de las partes incluidas.
JAVA_SOURCE	Valor de Sí o No. Si es Sí, incluya la fuente Java en JAR, WAR o EAR.
CLASSPATH_JARS	Directorio de vía de acceso de z/OS UNIX System Services utilizado para resolver dependencias de vía de acceso durante la construcción. Todos los jar ubicados en este directorio se utilizarán en la vía de acceso.
CLASSPATH_JARS_FILES	Nombres de archivos JAR y Class individuales que deben incluirse en la construcción. Ésta puede aparecer en forma de lista, tal como se ejemplifica a continuación: <code><property name="CLASSPATH_JARS_FILES" value="V2J4.jar,V2J3.jar" /></code>
ENCODING	La página de códigos ASCII o EBCDIC para JAVA. Se trata de la fuente JAVA de página de códigos que se almacena en el host de z/OS. Por ejemplo: <ul style="list-style-type: none"> • Para la página de códigos estándar JAVA ASCII debería ser ISO8859-1 • Para la página de códigos estándar JAVA EBCDIC debería ser IBM-1047
JAR_FILE_NAME EJB_NAME WAR_NAME EAR_NAME	Nombre de JAR, JAR EJB, WAR o EAR.
DEBUG_MODE	Establézcalo en 'on' (activo) para que Developer Toolkit no pueda eliminar ningún archivo de construcción del directorio WORKAREA. Resulta útil si necesita comprobar la estructura de una aplicación Java/J2EE construida.

Dependencias de CLASSPATH

La fuente Java dentro de una ARCHDEF puede tener dependencias de vía de acceso sobre otras bibliotecas o clase Java. Si estas dependencias se encuentran en componentes Java contenidos dentro de la misma estructura ARCHDEF, estas dependencias de vía de acceso se resuelven como parte de la construcción ARCHDEF (tanto si la modalidad de construcción es condicional como forzada).

No obstante, es posible que un componente ARCHDEF J2EE tenga dependencias de vía de acceso sobre JAR externos o incluso sobre miembros contenidos en otras ARCHDEF. En tal caso, el script de construcción J2EE asociado con la ARCHDEF puede controlar dependencias de vía de acceso con las siguientes palabras clave:

CLASSPATH_JARS

Se trata del nombre de directorio del sistema de archivos de z/OS UNIX System Services que podría incluir todos los archivos y clases JAR dependientes externos para dicha construcción ARCHDEF en particular.

Este directorio se puede actualizar con archivos CLASSPATH mediante la función Client Team "Cargar archivos jar" para copiar archivos JAR del cliente en el directorio de vía de acceso. También está disponible la función "Copiar archivo de SCLM en vía de acceso" para copiar archivos JAR almacenados en SCLM en el directorio de vía de acceso.

CLASSPATH_JARS_FILES

Esta palabra clave se puede utilizar para elegir selectivamente archivos Class o JAR individuales para utilizarlos en la vía de acceso. Si se utiliza esta palabra clave, los archivos JAR/class listados se recuperan desde SCLM o, si no se localizan en SCLM, se busca el directorio referenciado por CLASSPATH_JARS para su recuperación. Si se utiliza esta palabra clave, sólo se utilizarán aquellos archivos listados en la vía de acceso.

Scripts de ejemplo J2EE

El ejemplo siguiente muestra el script JAR:

```
<ANTXML>
<project name="JAVA Project" default="jar" basedir=". ">
<property name="env" environment="env" value="env"/>
<property name="SCLM_ARCHDEF" value="V2JAR1"/>
<property name="SCLM_ANTXML" value="BWBJAVA"/>
<property name="SCLM_BLDMAP" value="YES"/>
<property name="JAR_FILE_NAME" value="V2TEST.jar"/>
<property name="CLASSPATH_JARS" value="/var/rdz/CLASSPATH"/>
<property name="ENCODING" value="IBM-1047"/>
</ANTXML>
```

Figura 9. Muestra JAR de script de construcción J2EE

El ejemplo siguiente muestra el script WAR:

```
<ANTXML>
<project name="J2EE WAR Project" default="web-war" basedir=". ">
<property name="env" environment="env" value="env"/>
<property name="SCLM_ARCHDEF" value="SAMPLE5"/>
<property name="SCLM_ANTXML" value="BWBWEBA"/>
<property name="SCLM_BLDMAP" value="YES"/>
<property name="JAVA_SOURCE" value="YES"/>
<property name="J2EE_HOME" value="${env.J2EE_HOME}"/>
<property name="JAVA_HOME" value="${env.JAVA_HOME}"/>
<property name="CLASSPATH_JARS" value="/var/rdz/CLASSPATH"/>
<property name="ENCODING" value="IBM-1047"/>
<!-- Nombre de archivo WAR que debe crear este proceso de construcción -->
<property name="WAR_NAME" value="Sample5.war" />
<path id="build.class.path">
  <pathelement location="."/>
  <pathelement location="${J2EE_HOME}/lib/j2ee.jar"/>
  <pathelement location="${CLASSPATH_JARS}/jdom.jar"/>
<fileset dir="${CLASSPATH_JARS}" includes="**/*.jar, **/*.zip"/>
</path>
</ANTXML>
```

Figura 10. Muestra WAR de script de construcción J2EE

El ejemplo siguiente muestra el script EJB:

```
<ANTXML>
<project name="J2EE EJB Project" default="EJBBuild" basedir=". ">
<property name="env" environment="env" value="env"/>
<property name="SCLM_ARCHDEF" value="SAMPLE3"/>
<property name="SCLM_ANTXML" value="BWBEJBA"/>
<property name="SCLM_BLDMAP" value="NO"/>
<property name="J2EE_HOME" value="${env.J2EE_HOME}"/>
<property name="JAVA_HOME" value="${env.JAVA_HOME}"/>
<property name="CLASSPATH_JARS" value="/var/rdz/CLASSPATH"/>
<property name="ENCODING" value="IBM-1047"/>
<property name="EJB_NAME" value="DateService.jar"/>
<path id="build.class.path">
  <pathelement location="."/>
  <pathelement location="${J2EE_HOME}/lib/j2ee.jar"/>
  <pathelement location="${CLASSPATH_JARS}/jdom.jar"/>
<fileset dir="${CLASSPATH_JARS}" includes="**/*.jar, **/*.zip"/>
</path>
</ANTXML>
```

Figura 11. Muestra EJB de script de construcción J2EE

El ejemplo siguiente muestra el script EAR:

```
<ANTXML>
<project name="J2EE EAR Project" default="j2ee-ear" basedir=".">
  <property name="env" environment="env" value="env"/>
  <property name="SCLM_ARCHDEF" value="SAMPLE6"/>
  <property name="EAR_NAME" value="Sample6.ear"/>
  <property name="SCLM_ANTXML" value="BWBEARA"/>
  <property name="SCLM_BLDMAP" value="NO"/>
  <property name="J2EE_HOME" value="${env.J2EE_HOME}"/>
  <property name="JAVA_HOME" value="${env.JAVA_HOME}"/>
  <property name="CLASSPATH_JARS" value="/var/rdz/CLASSPATH"/>
  <path id="build.class.path">
    <pathelement location="."/>
    <pathelement location="${J2EE_HOME}/lib/j2ee.jar"/>
    <pathelement location="${CLASSPATH_JARS}/jdom.jar"/>
    <fileset dir="${CLASSPATH_JARS}" includes="**/*.jar, **/*.zip"/>
  </path>
  <target name="common">
    <echo message="BuildName: ${Ant.project.name}" />
    <echo message="BuildHome: ${basedir}" />
    <echo message="BuildFile: ${Ant.file}" />
    <echo message="BuildJVM: ${Ant.java.version}" />
  </target>
</ANTXML>
```

Figura 12. Muestra EAR de script de construcción J2EE

Esqueletos de construcción XML Ant JAVA/J2EE

Esta sección lista esqueletos de construcción Ant que se suministran en la biblioteca FEK.SFEKSAMV. Estos miembros de muestra se pueden copiar en el tipo SCLM J2EEBLD en la jerarquía SCLM que debe ser referenciada y utilizada por los scripts de construcción JAVA/J2EE. Los scripts de construcción JAVA/J2EE son archivos de variables de propiedades que superponen los archivos de esqueleto XML Ant.

Los esqueletos de construcción J2EE de muestra suministrados para construir un único JAR, proyecto SQLJ, JAR EJB, WAR, EAR o para el despliegue se pueden utilizar normalmente tal cual, sin necesidad de que el usuario efectúe ninguna personalización. No obstante, tenga en cuenta que es posible que algunos de los proyectos J2EE no se ajusten al modelo estándar, por lo que es posible que deba personalizar un poco los esqueletos XML Ant suministrados.

Nota: Los scripts de construcción JAVA/J2EE se pueden generar mediante la aplicación de cliente SCLM Developer Toolkit. Estos scripts de construcción utilizan un esqueleto XML Ant referenciado (tal como se ve a continuación) y una ARCHDEF en el proceso de construcción JAVA/J2EE.

Encontrará una descripción detallada de los scripts de construcción, los esqueletos Ant y ejemplos sobre el proceso de construcción JAVA/J2EE en la Guía del usuario de SCLM Developer Toolkit suministrada con el conector del cliente.

BWBJAVAA

Esqueleto de construcción JAVA XML Ant de muestra

Este esqueleto Ant lo utiliza un script de construcción Java para compilar varios programas Java y, opcionalmente, crear un archivo Java Archive (JAR) que tenga una estructura determinada por una ARCHDEF especificada.

BWBEJBA

Esqueleto de construcción EJB J2EE XML Ant de muestra

Este esqueleto Ant lo utiliza un script de construcción J2EE para compilar o construir un proyecto EJB que normalmente crearía un JAR EJB con estructura determinada por una ARCHDEF especificada.

BWBWEBA

Esqueleto de construcción WEB J2EE XML Ant de muestra

Este esqueleto Ant lo utiliza un script de construcción J2EE para compilar o construir un proyecto WEB que normalmente crearía un archivo WEB Archive (WAR).

BWBEARA

Esqueleto ensamblado EAR J2EE XML Ant de muestra

Este esqueleto Ant lo utiliza un script de construcción J2EE como un proceso de ensamblado como preparación para un despliegue de aplicación J2EE. El proceso genera archivos Enterprise Archive (EAR) que se pueden desplegar en un servidor de aplicación web como WebSphere Application Server.

BWBSQLB

Script de construcción Java/SQLJ de ejemplo

Este esqueleto Ant lo utiliza un script de construcción J2EE para compilar o construir un proyecto JAR que utilice SQLJ.

BWBSQLBE

Script de construcción EJB/SQLJ de ejemplo

Este esqueleto Ant lo utiliza un script de construcción J2EE para compilar o construir un proyecto EJB que utilice SQLJ.

BWBC9DTJ

Ejemplo de conversión de Cloud 9 a SCLM DT.

BWBDEPJ1

Ejemplo para actualizar archivos .ser de SQLJ en un JAR en tiempo de despliegue mediante db2sqljcustomize.

BWBDEPJ2

Ejemplo de db2sqljcustomize en el que la propiedad longname copiará el JAR especificado de las ubicaciones de grupo y tipo indicadas en SCLM al directorio de destino especificado por "dest".

BWBDEPJ3

Esta rutina de ejemplo personalizará los archivos .ser contenidos en los archivos JAR seleccionados para el despliegue mediante db2sqljcustomize.

BWBTRANX

Conversor de construcción SCLM de ejemplo para los mensajes de error de construcción SYSXML para COBOL.

Correlación de proyectos J2EE con SCLM

IBM SCLM Developer Toolkit proporciona la posibilidad de gestionar, construir y desplegar proyectos en SCLM. Esta sección describe cómo configurar la estructura de proyectos SCLM para que soporte el despliegue de aplicación distribuida como JAVA/J2EE.

Muchos proyectos JAVA/J2EE dan como resultado la creación de un archivo EAR ejecutable. Esta aplicación es un conjunto de proyectos, habitualmente aplicaciones EJB y Web y, dentro de los entornos IDE, normalmente se desarrollan como estructuras de proyecto individuales que están enlazadas con un proyecto EAR.

Este formato de estructura de varios proyectos no se coincide directamente con SCLM. Es decir, un proyecto SCLM no se puede enlazar con otro proyecto SCLM para proporcionar alguna forma de estructura de proyectos agregada. No obstante, SCLM proporciona un medio para soportar esta estructura de varios proyectos dentro de un único proyecto SCLM utilizando tipos.

Los proyectos SCLM se pueden definir con varios tipos de fuente. Cada tipo puede mantener un único proyecto IDE. Si intentáramos almacenar varios proyectos IDE Eclipse en SCLM sin ninguna forma de segregación, cada uno de los archivos .classpath y .project del proyecto se sobrescribirían, pues cada proyecto se ha añadido a SCLM. El uso de distintos tipos de fuente permite que estos archivos, y todos los demás asociados a dicho proyecto, se almacenen de forma segura dentro de SCLM.

Esta correlación haría que los proyectos IDE se almacenaran de forma independiente dentro de SCLM utilizando el tipo como diferenciador principal. Por ejemplo, EJB1 se almacena en el proyecto SCLM SCLMPRJ1 bajo el tipo EJB1. La utilización de esta estructura permite correlacionar la estructura de proyectos IDE con tipos independientes dentro del proyecto SCLM.

Notas:

1. Es posible que no sea necesario correlacionar un nombre de proyecto de IDE con el nombre del tipo SCLM; estos nombres existen independientemente el uno del otro.
2. Los nombres de tipo están restringidos a ocho caracteres; por ello, un proyecto IDE denominado "ProyectoUno" no podría tener el correspondiente nombre de tipo "ProyectoUno". En su lugar, puede utilizar "Proy1".

Por ello, es importante planificar la estructura de proyectos SCLM para adaptar la correlación de distintos proyectos basados en IDE en la única estructura de proyectos SCLM. Esto se debe a que, dentro de proyectos SCLM grandes, es importante añadir tipos de proyectos adicionales, ya que se requiere un cambio en la definición de proyectos SCLM, una reconstrucción de la definición de proyectos SCLM y la asignación de conjuntos de datos para los tipos nuevos.

Esta estructura no está restringida a proyectos de estilo J2EE; también puede aplicarse en cualquier situación en la que se estén desarrollando varios proyectos que sostengan algún tipo de dependencia entre sí.

Recomendaciones para correlaciones proyectos J2EE con SCLM

La siguiente lista proporciona recomendaciones para correlacionar proyectos J2EE (y otros) con SCLM:

- Identifique la composición del proyecto J2EE en cuanto a EJB, aplicaciones web, etc., de forma que pueda utilizarse para planificar la estructura de proyectos SCLM.
- Para cada uno de los componentes de proyecto IDE J2EE, cree un tipo correspondiente en el proyecto SCLM. Resulta conveniente proporcionar algún tipo de convenio de denominación que tenga sentido para darle soporte. Si bien es posible denominar los proyectos IDE sin tener en cuenta el tipo SCLM, las correlaciones facilitarán la administración.
- Puesto que los requisitos del proyecto pueden cambiar, cree definiciones de tipo adicionales para permitir la adición uniforme de otros componentes, tales como EJB adicionales. Se pueden anticipar servicios adicionales mediante la estructura de tipo.
- La construcción de tipo soporta la correlación de varios proyectos IDE en un único proyecto SCLM. También resulta útil aplicar algunas estructuras empaquetadas que tengan en cuenta la definición de tipo para dicho proyecto.
- Las convenciones de empaquetado de estilo Java también se pueden definir a nivel de proyecto para evitar la posibilidad de colisiones de denominación de fuente.
- Si el IDE está estructurado con varios proyectos, es recomendable duplicar esta estructura en SCLM utilizando el tipo.

El uso de varios tipos SCLM para almacenar proyectos IDE individuales también está relacionado con la operación de la estructura ARCHDEF para la construcción de estos proyectos IDE.

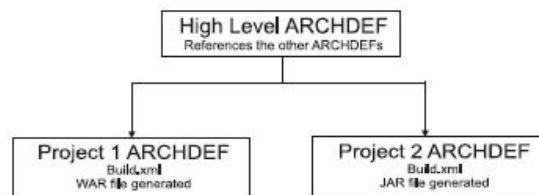


Figura 13. Jerarquía de construcción SCLM

El archivo ARCHDEF contiene la lista de archivos que forman una construcción. En un contexto J2EE, una construcción puede tener como consecuencia que un archivo EAR se componga de varios archivos WAR y JAR. Este aislamiento de proyectos es parecido a la estructura de tipo que define el proyecto en SCLM. Si se tiene una ARCHDEF de nivel alto que hace referencia a esas "partes" que forman

la construcción, es posible tener un entorno de construcción estructurado. Esto está relacionado con la definición efectiva de la estructura del proyecto al definir los tipos en SCLM.

La definición del proyecto de forma estructurada también permite:

- La migración de archivos de un tipo de proyecto SCLM o ARCHDEF a un proyecto IDE sin la necesidad de conocer las partes individuales.
- La estructura ARCHDEF basada en definición de tipo también permite que las dependencias del proyecto se correlacionen de forma más efectiva. Es habitual que los proyectos IDE hagan referencia a otros proyectos IDE del espacio de trabajo. El uso de la palabra clave SCML INCL en las ARCHDEF apoya esta noción porque otros proyectos IDE, referenciados por otras ARCHDEF, se pueden incluir anidando las ARCHDEF dentro de ARCHDEF de nivel superior.

Al construir aplicaciones con referencias o dependencias a otros objetos de construcción como JAR, otros proyectos y otras clases, existen los múltiples métodos siguientes:

1. Incluir la referencia a JAR mediante la sentencia INCLD en la ARCHDEF. Esto construirá la aplicación con la referencia a la biblioteca en el paquete de construcción final.
2. Incluir el proyecto IDE como una ARCHDEF de proyecto SCLM INCL anidada.
3. Incluir el JAR dependiente en el directorio CLASSPATH.
 - Si el proyecto IDE se refiere a un JAR pero no se espera que forme parte del paquete de construcción final, el archivo de biblioteca se puede copiar en el sistema CLASSPATH utilizando el servicio Subir archivos JAR de Developer Toolkit. Ello dará como resultado que este servicio esté disponible desde un proyecto SCLM distinto. Durante el tiempo de construcción, se resolverán las referencias del proyecto IDE al archivo JAR. No obstante, el JAR debe estar disponible en una sentencia PATH durante el tiempo de ejecución.
 - Hacer referencia a un JAR que se encuentra en el mismo proyecto SCLM utilizando la propiedad CLASSPATH_JARS_FILES en el script de construcción.

Despliegue de SCLM Developer Toolkit

SCLM Developer Toolkit proporciona varias funciones de despliegue. Puede desplegar archivos Enterprise Archive (EAR) en cualquier WebSphere Application Server (WAS). Además, cualquier componente construido o controlado por SCLM Developer Toolkit se puede distribuir utilizando un script de despliegue personalizable. Se suministran scripts de muestra que se pueden utilizar para copiar un EAR en un sistema principal remoto utilizando los mandatos "Secure copy" (SCP) y "Secure FTP" (SFTP).

En el núcleo del despliegue, básicamente existen dos scripts; el primer tipo de script, el que modifica el usuario, es el script de propiedades y contiene una lista de parámetros para la operación de despliegue. El segundo es el script de acción que contiene los pasos necesarios para ejecutar la operación de despliegue.

El despliegue se inicia desde el conector del cliente de SCLM Developer Toolkit y el tipo de despliegue se selecciona pulsando el botón pertinente en la pantalla de despliegue. La acción de despliegue que se elija influirá en lo que se llenará en el script de propiedades. Para la mayoría de scripts, existe una propiedad denominada SCLM_ANTXML que contiene el nombre del miembro del

correspondiente script de acción. Developer Toolkit toma el script de propiedades generado y lo superpone al script de acción, antes de invocar al script de acción resultante.

A continuación encontrará una lista de scripts de acción de despliegue Ant que se suministran con la biblioteca FEK.SFEKSAMV. Estos miembros de muestra se pueden copiar en SCLM tipo J2EEBLD, en la jerarquía SCLM que debe ser referenciada y utilizada por los scripts de propiedades generados. Los scripts de propiedades generados son archivos de variables de propiedades que se superponen a los scripts de acción de despliegue XML Ant referenciados más abajo. Estos scripts deben almacenarse con un lenguaje de tipo de texto, como TEXT o J2EEPART.

Miembro	Descripción
BWBDEPLA	Despliegue de EAR en WAS.
BWBRDEPL	Despliegue de EAR en WAS remoto.
BWBSCOPY	Despliegue de "Secure copy". Copia un objeto de construcción desde un host a otro mediante SCP.
BWBSFTP	Despliegue de "Secure FTP". Copia un objeto de construcción desde un host a otro utilizando SFTP.

Para que estos scripts de construcción puedan utilizarlos varios grupos, el administrador debe construir y promover los scripts al nivel de grupo superior disponible en el proyecto.

Se produce un proceso ligeramente distinto en función de los tipos de script que se generen.

Despliegue de WebSphere Application Server (WAS)

Para el despliegue de WebSphere Application Server (WAS), la propiedad SCLM_ANTXML no apunta hacia un script de acción Ant; en su lugar, hace referencia a un script de acción JACL. De forma alternativa, puede utilizar la herramienta wsadmin que se suministra con WAS en z/OS.

La herramienta wsadmin requiere un script JACL para guiar el proceso de despliegue. Si está utilizando este método de despliegue, el script JACL debe crearse como un archivo ASCII en un directorio de z/OS UNIX antes de poder invocar al proceso de despliegue.

La personalización de Developer for System z facilita un script JACL (ASCII) de muestra como /etc/rdz/sclmdt/CONFIG/scripts/deploy.jacl (donde /etc/SCLMDT es el directorio etc predeterminado para SCLM Developer Toolkit).

Podrá encontrar ejemplos de JACL adicionales en la documentación de WebSphere Application Server (WAS).

Las ubicaciones de directorio de la herramienta wsadmin (wsadmin.sh) y del script JACL (deploy.jacl) se pueden configurar en la página de preferencias bajo **Equipo > Preferencias de SCLM > Opciones de script de construcción**. El cliente SCLMDT se utiliza para generar un script de despliegue contra el que se puede construir a continuación. (El proceso de despliegue lo desencadena una solicitud de función contra el script de despliegue que está almacenado en el tipo SCLM J2EEBLD).

Los scripts de acción de muestra que deben almacenarse en el tipo SCLM J2EEBLD para el despliegue de WAS o el despliegue de WAS remoto son BWBDEPLA y BWBRDEPL.

Despliegue de SCLM en UNIX System Services

SCLM Developer Toolkit facilita permite desplegar todos los archivos que están almacenados en el repositorio de SCLM en el sistema de archivos de z/OS UNIX System Services en el mismo LPAR. Se trata de una forma sencilla de desplegar un objeto construido por SCLM en un entorno donde se puede ejecutar o desplegar en un host remoto utilizando el Despliegue seguro descrito más abajo.

No existe ningún script de acción de muestra para esta acción. Seleccione los miembros SCLM y utilice el botón Incluir miembros SCLM para generar el script de propiedades necesario. De esta manera se copian los archivos de una ubicación de SCLM seleccionada en un directorio especificado en el sistema de archivos de z/OS UNIX System Services. Este directorio debe existir previamente o se producirá un error.

Despliegue seguro

Esta opción proporciona una forma de copiar objetos desplegables en un host remoto utilizando los mandatos "Secure copy" (SCP) y "Secure FTP" (SFTP). Utilizando una combinación del script de propiedades de Despliegue seguro e Incluir miembros SCLM, se pueden seleccionar los archivos necesarios en la jerarquía SCLM, copiarlos en una ubicación del sistema de archivos de z/OS UNIX System Services y, finalmente, copiarlos en la máquina destino desde dicho sistema de archivos de z/OS UNIX System Services mediante los mandatos "Secure copy" (SCP) y "Secure FTP" (SFTP).

Los scripts de acción de muestra que deben almacenarse en el tipo SCLM J2EEBLD para el despliegue seguro son BWBSCOPY y BWBSFTP.

Nota: Deberá pedir, instalar y configurar IBM Ported Tools for z/OS para dar soporte al despliegue seguro. Consulte la *Guía de planificación de host de IBM Rational Developer for System z* (GI11-7839) para obtener información acerca de cómo obtener Ported Tools. La instalación y la personalización de este producto no se describen en este manual.

IBM Ported Tools for z/OS proporciona lo siguiente:

- scp para copiar archivos entre redes. Es una alternativa a rcp.
- sftp para transferencias de archivos mediante un transporte ssh cifrado. Se trata de un programa de transferencia de archivos interactivo parecido a ftp.

Autenticación de clave pública

La autenticación de clave pública constituye una alternativa al inicio de sesión interactivo que se puede automatizar como parte de la operación de despliegue seguro de Developer Toolkit.

Para que la autenticación de clave pública funcione de la forma deseada, puede utilizar un ID de usuario sustituto para el despliegue o puede configurar cada usuario al que desee suministrar posibilidades de despliegue.

Para obtener instrucciones acerca de cómo configurar la autenticación basada en clave automatizada utilizando ssh-agent y ssh-add, consulte el manual *IBM Ported*

Tools for z/OS User's Guide. Para obtener información acerca del ID de usuario sustituto de SCLM Developer Toolkit, consulte Capítulo 4, "Seguridad de SCLM", en la página 47.

Otras opciones de despliegue

También puede crear sus propios scripts Ant para efectuar despliegues de varias formas distintas. En sus scripts, si utiliza el código <exec> de Ant, podrá invocar a todos los programas que estén disponibles en el sistema de archivos de z/OS UNIX System Services. Utilizando este método, los scripts de construcción pueden llamar a otros programas, como por ejemplo el protocolo de transferencia de archivos (FTP), para efectuar el despliegue. Para obtener más información sobre la creación de scripts de Ant, consulte la documentación sobre Ant en línea en <http://ant.apache.org/>.

Opciones de almacenamiento ASCII o EBCDIC

Los archivos fuente transferidos desde el conector de SCLM Developer Toolkit se pueden almacenar en SCLM como ASCII o EBCDIC.

En general, todas las fuentes en SCLM se almacenan en EBCDIC para poder ser visualizadas y editadas directamente desde ISPF/SCLM en z/OS. Si no desea examinar ni editar código directamente desde el host, es posible que prefiera almacenarlo directamente (es decir, transferido como binarios) donde se almacena la fuente en SCLM utilizando la página de códigos ASCII/UNICODE original del cliente. Esto favorece el rendimiento de los proyectos grandes que se almacenan e importan en SCLM y de las construcciones JAVA/J2EE, porque no se realizará una conversión de ASCII a EBCDIC.

SCLM Developer Toolkit determina si un archivo se transfiere como binario o si se produce una conversión de ASCII a EBCDIC comprobando el lenguaje SCLM asociado a cada archivo o miembro. A continuación, SCLM Developer Toolkit comprueba si dicho lenguaje SCLM tiene una entrada en el archivo TRANSLATE.conf con una palabra clave TRANLANG.

Conversores de lenguaje ASCII/EBCDIC

Tabla 4. Conversores de lenguaje SCLM y ASCII/EBCDIC

Conversor de lenguaje SCLM	Descripción
JAVA	Miembros fuente Java almacenados como EBCDIC. Creados utilizando BWBTRAN1 de muestra.
SQLJ	Miembros SQLJ almacenados como EBCDIC. Creados utilizando una muestra (BWBTRANS).
JAVABIN	Miembros fuente Java almacenados como ASCII. Creados utilizando BWBTRAN1 de muestra.
J2EEPART	Todos los archivos J2EE en los que no es necesario efectuar ningún análisis y se almacenen como EBCDIC. Creados utilizando BWBTRAN1 de muestra.
J2EEBIN	Todos los archivos J2EE en los que no es necesario efectuar ningún análisis y se almacenen como archivos binarios o ASCII. Creados utilizando BWBTRAN1 de muestra.
SQLJ	Miembros fuente SQLJ almacenados como EBCDIC. Creados utilizando BWBTRANS de muestra.
SQLJBIN	Miembros fuente SQLJ almacenados como ASCII. Creados utilizando BWBTRANS de muestra.

Tabla 4. Conversores de lenguaje SCLM y ASCII/EBCDIC (continuación)

Conversor de lenguaje SCLM	Descripción
TEXT	Conversor TEXT predeterminado en el que no es necesario efectuar ningún análisis y se almacena como EBCDIC. Creados utilizando BWBTRAN1 de muestra.
BINARY	Conversor de lenguaje binario predeterminado en el que no se necesita efectuar ningún análisis. Creados utilizando BWBTRAN1 de muestra.

Se presupone que el uso predeterminado será la conversión ASCII/EBCDIC. Esto significa que los archivos examinados y editados en el conector de Eclipse también se pueden examinar y editar directamente en el host desde ISPF/SCLM.

El uso de ASCII (transferido como binario) es recomendable para la migración o importación de proyectos y el rendimiento de la construcción, ya que los archivos no precisan ninguna conversión. Sólo resulta adecuado si no se necesita la edición en ISPF/SCLM.

En función del conversor de lenguaje SCLM utilizado, la fuente se podrá construir con ASCII o EBCDIC.

Para la utilización en varias plataformas, los archivos desplegables tales como JAR, WAR y EAR se construyen de forma que todos los objetos incluidos sean del tipo ASCII, independientemente de que alguna de las fuentes se almacene como EBCDIC.

Nota sobre la construcción JAVA/J2EE: Si el código fuente Java se almacena como ASCII, el script de construcción debe especificar la página de códigos ASCII utilizando la variable de propiedad ENCODING para compilar correctamente el código fuente Java.

Por ejemplo:

```
<property name="ENCODING" value="ISO8859-1"/>
```

El script de Ant llamado utilizará el mandato Java con ENCODING=ISO8859-1 para compilar la fuente ASCII. La página de códigos ENCODING predeterminada es la página de códigos EBCDIC IBM-1047.

\$GLOBAL

Como parte del proceso de construcción JAVA/J2EE, se necesita información adicional para realizar las construcciones de forma satisfactoria. Puesto que las construcciones se realizan en z/OS UNIX System Services, se necesita información tal como la ubicación del producto Java, la ubicación del producto Ant y la ubicación del área de trabajo y de los archivos de configuración de SCLM Developer Toolkit.

Además, es posible que se necesiten diferentes versiones de Ant o Java para diferentes grupos de desarrollo de SCLM, por lo que con este fin el miembro \$GLOBAL puede ser específico de grupo. Las variables de entorno establecidas en \$GLOBAL pueden sobrescribirse con valores de variables de script de construcción específicos.

Nota: Cuando se utiliza el archivo de configuración ant.conf, el valor JAVA_HOME especificado alterará temporalmente todos los valores de JAVA_BIN especificados en \$GLOBAL para compilaciones de proyecto Java/J2EE. Esto no

es cierto si la configuración Ant se realiza en `rsed.envvars`, que es lo que se describe en la *Guía de configuración de host de Rational Developer for System z* SC11-3660-03 (SC23-7658).

Se suministra un miembro de muestra `BWBGLOB` en la biblioteca `FEK.SFEKSAMV`. Este miembro de muestra debe copiarse en el tipo `SCLM J2EEBLD`, en la jerarquía de `SCLM` como miembro `$GLOBAL`, y debe guardarse con un lenguaje de no análisis válido como `TEXT` (tal como se suministra en el conversor de lenguaje `FLM@TEXT` en la biblioteca `SISPMACS`).

El miembro `$GLOBAL` actualmente pone a su disposición la siguiente información sobre los procesos de construcción `JAVA/J2EE`:

Tabla 5. Variables `$GLOBAL`

Variable	Descripción
ANT_BIN	Vía de acceso de directorio de sistema de archivos de z/OS UNIX System Services del tiempo de ejecución de Ant Ejemplo: <code><property name="ANT_BIN" value="/usr/lpp/apache-Ant-1.6.0/bin/Ant"/></code>
JAVA_BIN	Vía de acceso de directorio de sistema de archivos de z/OS UNIX System Services del tiempo de ejecución/compilación de Java Ejemplo: <code><property name="JAVA_BIN" value="/usr/lpp/java/5.0/bin"/></code>
_SCLMDT_WORK_HOME	La ubicación del directorio <code>WORKAREA</code> de <code>SCLM Developer Toolkit</code> Ejemplo: <code><property name="_SCLMDT_WORK_HOME" value="/var/rdz"/></code>
_SCLMDT_CONF_HOME	Ubicación del directorio <code>CONFIG</code> de <code>SCLM Developer Toolkit</code> Ejemplo: <code><property name="_SCLMDT_CONF_HOME" value="/etc/rdz/sclmdt"/></code>
CLASSPATH_JARS	Directorio de vía de acceso de clases del sistema de archivos de z/OS UNIX System Services que se utiliza para compilaciones <code>JAVA</code> . Todos los <code>jar</code> ubicados en este directorio se utilizarán en la vía de acceso. Ejemplo: <code><property name="CLASSPATH_JARS" value="/var/rdz/CLASSPATH"/></code>
TRANTABLE	Archivo <code>VSAM</code> que contiene las conversiones de nombre largo/corto Ejemplo: <code><property name="TRANTABLE" value="FEK.#CUST.LSTRANS.FILE"/></code>
DEBUG_MODE	Establézcalo en 'on' (activo) si desea que <code>Developer Toolkit</code> no elimine archivos de construcción <code>Java/J2EE</code> del sistema de archivos de z/OS UNIX System Services. Resulta útil si desea ver la estructura de las salidas de construcciones en el sistema de archivos <code>USS</code> para efectuar depuraciones.

Si estas variables deben establecerse para todos los niveles de grupos en el proyecto `SCLM`, resulta una buena práctica crear un miembro `$GLOBAL` único en el nivel superior de la jerarquía. Cuando se ejecute el conversor de construcciones

JAVA/J2EE, buscará la jerarquía del nivel de grupo que realiza la construcción y utilizará el primer \$GLOBAL que encuentre en el tipo J2EEBLD.

Nota: El miembro \$GLOBAL debe almacenarse como un miembro SCLM guardado válido de forma que pueda efectuarse esta búsqueda en la jerarquía.

Si se necesitan valores distintos en diferentes grupos de desarrollo, se puede crear un miembro \$GLOBAL en cada uno de los grupos de desarrollo.

TRANSLATE.conf

El archivo TRANSLATE.conf proporciona palabras clave para determinar la forma en la que se almacenan los códigos dentro de SCLM. El archivo de configuración contiene palabras clave que determinan cómo se transfieren los archivos al host en función de su definición de lenguaje. Las palabras clave determinan si los archivos de un tipo de lenguaje en concreto son binarios, si se transfieren o almacenan o si la fuente basada en texto debe permanecer en formato ASCII en lugar de seguir la conversión predeterminada de ASCII en EBCDIC.

Además, las definiciones de lenguaje de SCLM controlan si los archivos con nombres largos deben convertirse en nombres de host cortos válidos para poder almacenarse en SCLM. Esta correlación de nombres largos a cortos la controla el VSAM de conversión de nombres largos/cortos de SCLM.

TRANSLATE.conf está ubicado en /etc/rdz/sc1mdt/CONFIG. Puede editar el archivo con el mandato TSO OEDIT.

El siguiente ejemplo muestra el código TRANSLATE.conf, que debe personalizarse para que se adapte a su entorno de sistema. Las líneas de comentarios empiezan por un asterisco (*).

```
*=====
* compartimiento cruzado
TRANVRLS = NO
*=====
* página de códigos
CODEPAGE ASCII = IS08859-1
CODEPAGE EBCDIC = IBM-1047
*=====
* conversión ascii/ebcdic
TRANLANG JAVABIN
TRANLANG J2EEBIN
TRANLANG J2EEOBJ
TRANLANG TEXTBIN
TRANLANG BINARY
TRANLANG DOC
TRANLANG XLS
*=====
* conversión de nombre largo/corto
LOGLANG JAVA
LOGLANG SQLJ
LOGLANG J2EEPART
LOGLANG JAVABIN
LOGLANG J2EEBIN
LOGLANG J2EEOBJ
LOGLANG DOC
LOGLANG XLS
```

Figura 14. TRANSLATE.conf - Archivo de configuración de conversión ASCII/EBCDIC de SCLMDT

Las palabras clave siguientes son válidas dentro del archivo TRANSLATE.conf:

code page ASCII

Indica las páginas de códigos ASCII que deben utilizarse en la conversión. El valor predeterminado es IS08859-1.

Debe haber una directiva de página de códigos para ASCII y EBCDIC para que SCLM Developer Toolkit determine cómo convertir los archivos que se están transfiriendo.

Code page EBCDIC

Indica las páginas de códigos EBCDIC que deben utilizarse en la conversión. El valor predeterminado es IBM-1047.

Debe haber una directiva de página de códigos para ASCII y EBCDIC para que SCLM Developer Toolkit determine cómo convertir los archivos que se están transfiriendo.

TRANVRLS

Indica si el conjunto de datos VSAM de conversión de nombres largos/cortos se puede compartir entre sistemas. El valor predeterminado es NO. Los únicos valores válidos son YES y NO.

SCLM utiliza Record Level Sharing (RLS) de VSAM para permitir que se pueda compartir el conjunto de datos VSAM y mantener la integridad en un entorno compartido. Los conjuntos de datos VSAM deben definirse con la STORAGECLASS correcta para que RLS (Record Level Sharing) pueda utilizarlos. Consulte el manual *DFSMS(TM) Using Data Sets (SC26-7410)* para obtener más información sobre RLS.

TRANLANG

Indica qué tipos de lenguaje SCLM no necesitan ninguna conversión ASCII/EBCDIC (los archivos se transferirán en binario al host).

Tenga en cuenta que no existe ningún signo igual (=) en esta directiva para separar la palabra clave TRANLANG y el nombre del Conversor de lenguaje (ficticio).

LONGLANG

Determina qué tipos de lenguaje SCLM precisan una conversión de nombre largo a nombre corto. La conversión de nombre largo a nombre corto implica que el archivo del nombre largo del Cliente (incluida la estructura de paquetes del directorio) se correlacionará con un nombre de miembro de host válido de ocho caracteres y se almacenará en SCLM utilizando dicho nombre corto de host convertido.

Si no se especifica el lenguaje SCLM en la palabra clave LONGLANG, se asume que el archivo de cliente ya está dentro del formato de nombre corto del host (8 caracteres o menos) y se almacena como tal.

Tenga en cuenta que no existe ningún signo igual (=) en esta directiva para separar la palabra clave LONGLANG y el nombre del lenguaje SCLM.

Nota: Los valores establecidos en el archivo TRANSLATE.conf se pueden alterar temporalmente en un nivel de proyecto SITE y SCLM, tal como se describe en "SITE y opciones específicas de proyecto".

SITE y opciones específicas de proyecto

Se ha suministrado un recurso que permite generar determinados valores a un nivel de instalación SITE o a un nivel de proyecto SCLM específico. Las opciones que actualmente se pueden configurar son las siguientes:

- Entrada de código de cambio obligatoria
- Desactivación de construcciones y promociones en primer plano
- Especificación del sistema de aprobación de paquetes. Actualmente, IBM Breeze for SCLM es el sistema de aprobación soportado.
- Definición de tarjetas de trabajo de construcción por lotes, promoción y migración
- Alteración temporal de valores en el archivo de configuración TRANSLATE.conf
- Restricción de filtros de lista de proyectos
- Definición de valores predeterminados para idiomas bidireccionales (bidi)

Se pueden establecer todas estas opciones o ninguna de ellas. Si no se establecen, los programas adoptarán las predeterminadas. Algunas de estas opciones se pueden establecer en el archivo SITE.conf mientras que otras se pueden establecer en un nivel específico de proyecto SCLM. De forma alternativa, puede no haber ningún archivo específico SITE-specific y las opciones se pueden establecer sólo en un nivel de proyecto SCLM. Para tarjetas de trabajo, puede alterar temporalmente la información de la tarjeta de trabajo utilizando su propia tarjeta de trabajo especificada e introducida a través del IDE.

Este recurso se activa creando el archivo SITE.conf en el directorio z/OS UNIX /etc/rdz/sc1mdt/CONFIG/PROJECT/ (donde /etc/rdz/sc1mdt es el directorio etc predeterminado para SCLM Developer Toolkit). Este directorio se crea durante la personalización de Developer for System z.

Se suministra un archivo SITE.conf de muestra en el directorio /usr/lpp/rdz/samples/. Copie este directorio y las directivas para que se adapten a sus requisitos. Puede editar el archivo con el mandato TSO OEDIT. El ejemplo siguiente muestra el archivo de configuración SITE.conf.

```

* Opción específica SITE de SCLM Developer Toolkit
*
* ¿El proceso aprobador SCM se aplica a este proyecto?
BUILDAPPROVER=NONE
PROMOTEAPPROVER=NONE
*
* ¿Es obligatoria la entrada de código de cambio en la incorporación?
CCODE=N
*
*
* Para permitir sólo la promoción por definición de arquitectura,
* establezca el valor PROMOTEONLYFROMARCHDEF en Y
PROMOTEONLYFROMARCHDEF=N
*
* ¿Se permiten construcciones/promociones en línea o en primer plano para este proyecto?
FOREGROUNDBUILD=Y
FOREGROUNDPROMOTE=Y
*
* Tarjeta de trabajo predeterminada de construcción por lotes
BATCHBUILD1=//SCLMBILD JOB (#ACCT),'SCLM BUILD',CLASS=A,MSGCLASS=X,
BATCHBUILD2=//          NOTIFY=&SYSUID,REGION=512M
BATCHBUILD3=//*
BATCHBUILD4=//*
*
* Tarjeta de trabajo predeterminada de promoción por lotes
BATCHPROMOTE1=//SCLMPROM JOB (#ACCT),'SCLM PROMOTE',CLASS=A,MSGCLASS=X,
BATCHPROMOTE2=//          NOTIFY=&SYSUID,REGION=128M
BATCHPROMOTE3=//*
BATCHPROMOTE4=//*
*
* Tarjeta de trabajo predeterminada de migración por lotes
BATCHMIGRATE1=//SCLMMIGR JOB (#ACCT),'SCLM MIGRATE',CLASS=A,MSGCLASS=X,
BATCHMIGRATE2=//          NOTIFY=&SYSUID,REGION=128M
BATCHMIGRATE3=//*
BATCHMIGRATE4=//*
*
* Tarjeta de trabajo predeterminada de despliegue por lotes
BATCHDEPLOY1=//SCLMDPLY JOB (#ACCT),'SCLM DEPLOY',CLASS=A,MSGCLASS=X,
BATCHDEPLOY2=//          NOTIFY=&SYSUID,REGION=128M
BATCHDEPLOY3=//*
BATCHDEPLOY4=//*
*
* Distintivo de seguridad BUILD para llamada de seguridad SAF/RACF y
* posible conmutación de ID sustituto
BUILDSECURITY=N
*
* Si se establece el distintivo de lista de proyectos en N, los usuarios
* no podrán seleccionar * como filtro de proyecto. Así se podrán evitar
* búsquedas de catálogo para todos los proyectos SCLM.
*
PROJECTLISTALL=Y

```

Figura 15. Valor de proyecto SCLM específico de SITE de muestra

También es posible tener valores de configuración específicos de proyecto que se utilicen para configurar un único proyecto SCLM. Éstos alterarán temporalmente los valores específicos SITE si existe un SITE.conf. Si desea establecer valores específicos de proyecto, deberá crear un archivo denominado <project>.conf en el directorio /PROJECT, donde <project> es el nombre del proyecto SCLM (no sensible a mayúsculas y minúsculas).

Se suministra un archivo de configuración de proyecto de muestra en el directorio /usr/lpp/rdz/samples/ como archivo SCLMproject.conf. Copie esta muestra en el directorio PROJECT utilizando el nombre de destino correcto y personalice las directivas para que se adapten a sus requisitos.

Puede editar el archivo con el mandato TSO OEDIT. El ejemplo siguiente muestra el código de Configuración de proyecto.

```
* Opción específica de proyecto de SCLM Developer Toolkit
*
* ¿El proceso aprobador SCM se aplica a este proyecto?
BUILDAPPROVER=BREEZE
PROMOTEAPPROVER=BREEZE
*
* ¿Es obligatoria la entrada de código de cambio en la incorporación?
CCODE=Y
*
* ¿Se permiten construcciones/promociones en línea o en primer plano para este proyecto?
FOREGROUNDBUILD=N
FOREGROUNDPROMOTE=N
*
* Tarjeta de trabajo predeterminada de construcción por lotes
BATCHBUILD1=//SCLMBILD JOB (#ACCT),'SCLM BUILD',CLASS=A,MSGCLASS=X,
BATCHBUILD2=//          NOTIFY=&SYSUID,REGION=512M
BATCHBUILD3=//*
BATCHBUILD4=//*
*
* Tarjeta de trabajo predeterminada de promoción por lotes
BATCHPROMOTE1=//SCLMPROM JOB (#ACCT),'SCLM PROMOTE',CLASS=A,MSGCLASS=X,
BATCHPROMOTE2=//          NOTIFY=&SYSUID,REGION=128M
BATCHPROMOTE3=//*
BATCHPROMOTE4=//*
*
* Distintivo de seguridad BUILD para llamada de seguridad SAF/RACF y
* posible conmutación de ID sustituto
BUILDSECURITY=N
* Distintivo de seguridad PROMOTE para llamada de seguridad SAF/RACF
* y posible conmutación de ID sustituto
PROMOTESECURITY=N
* Distintivo de seguridad J2EE DEPLOY para llamada de seguridad
* SAF/RACF y posible conmutación de ID sustituto
DEPLOYSECURITY=N
```

Figura 16. Valor de proyecto SCLM específico de PROJECT de muestra

Todas las opciones listadas son opcionales. Se utilizan los valores predeterminados si no se especifica nada en SITE.conf ni en project.conf.

Tabla 6. Opciones SITE/Project

BUILDAPPROVER=approval product/NONE	Especifique el nombre del producto de aprobación utilizado para el proceso de construcción. Actualmente, el único producto soportado es IBM Breeze for SCLM, que se selecciona con la palabra clave BREEZE. El valor predeterminado es NONE.
PROMOTEAPPROVER=approval product/NONE	Especifique el nombre del producto de aprobación utilizado para el proceso de promoción. Actualmente, el único producto soportado es IBM Breeze for SCLM. Si PROMOTEAPPROVER se establece en BREEZE, los campos específicos de Breeze se visualizarán durante una promoción. El valor predeterminado es NONE.
CCODE=N/Y	Especifique Y para que la entrada de código de cambio en la incorporación sea un campo obligatorio. El valor predeterminado es N para que la entrada de código de cambio no sea obligatoria.
FOREGROUNDBUILD=Y/N	Especifique N para restringir construcciones en primer plano. El valor predeterminado es Y, es decir: las construcciones en primer plano están permitidas.

Tabla 6. Opciones SITE/Project (continuación)

FOREGROUND_PROMOTE=Y/N	Especifique N para restringir promociones en primer plano. El valor predeterminado es Y , es decir: las promociones de primer plano están permitidas.
BATCHBUILD1=Job card 1 BATCHBUILD2=Job Card 2 BATCHBUILD3=Job Card 3 BATCHBUILD4=Job Card 4	Establezca una tarjeta de trabajo por lotes predeterminada para el proceso de construcción. Distintos proyectos pueden utilizar diferentes códigos de cuenta o clases de trabajo para que la opción de especificar tarjetas de trabajo específicas de proyecto permita este caso de ejemplo.
BATCHPROMOTE1=Job card 1 BATCHPROMOTE2=Job card 2 BATCHPROMOTE3=Job card 3 BATCHPROMOTE4=Job card 4	Establezca un trabajo por lotes predeterminado para el proceso de promoción. Distintos proyectos pueden utilizar diferentes códigos de cuenta o clases de trabajo para que la opción de especificar tarjetas de trabajo específicas de proyecto permita este caso de ejemplo.
BATCHMIGRATE1=Job card 1 BATCHMIGRATE2=Job card 2 BATCHMIGRATE3=Job card 3 BATCHMIGRATE4=Job card 4	Establezca un trabajo por lotes predeterminado para el proceso de migración. Distintos proyectos pueden utilizar diferentes códigos de cuenta o clases de trabajo para que la opción de especificar tarjetas de trabajo específicas de proyecto permita este caso de ejemplo.
BUILDSECURITY=Y/N	Especifique Y para invocar a la llamada de seguridad SAF/RACF para el paso de construcción y, posiblemente, efectuar una conmutación de ID sustituto. Para obtener más información, consulte Capítulo 4, “Seguridad de SCLM”, en la página 47.
PROMOTESECURITY=Y/N	Especifique Y para invocar a la llamada de seguridad SAF/RACF para el paso de promoción y, posiblemente, efectuar una conmutación de ID sustituto. Para obtener más información, consulte Capítulo 4, “Seguridad de SCLM”, en la página 47.
DEPLOYSECURITY=Y/N	Especifique Y para invocar a la llamada de seguridad SAF/RACF para el paso de despliegue y, posiblemente, efectuar una conmutación de ID sustituto. Para obtener más información, consulte Capítulo 4, “Seguridad de SCLM”, en la página 47.
ASCII=ASCII codepage	Especifique la página de códigos ASCII para alterar temporalmente la página de códigos ASCII especificada en TRANSLATE.conf. Por ejemplo: ASCII=UTF-8
EBCDIC=EBCDIC codepage	Especifique la página de códigos EBCDIC para alterar temporalmente la página de códigos EBCDIC especificada en TRANSLATE.conf. Por ejemplo: EBCDIC=IBM-420
TRANLANG=SCLM Language	Especifique un parámetro TRANLANG que deba añadirse a la lista de parámetros TRANLANG especificada en TRANSLATE.conf. Por ejemplo: TRANLANG=DOC
NOTRANLANG=SCLM Language	Utilice la palabra clave NOTRANLANG para eliminar un TRANLANG ya especificado en la lista de permisibles para este proyecto SCLM tal como se ha especificado en TRANSLATE.conf. Por ejemplo: NOTRANLANG=JAVA
LOGLANG=SCLM Language	Especifique un parámetro LOGLANG que deba añadirse a la lista de parámetros LOGLANG especificada en TRANSLATE.conf. Por ejemplo: LOGLANG=DOC

Tabla 6. Opciones SITE/Project (continuación)

NOLONGLANG=SCLM Language	<p>Utilice la palabra clave NOLONGLANG para eliminar un LONGLANG ya especificado en la lista de permisibles para este proyecto SCLM tal como se ha especificado en TRANSLATE.conf. Por ejemplo:</p> <p>NOLONGLANG=COBOL</p>
BIDIPROP=LANG=SCLM Language/* TextOrient=LTR/RTL TextType=Visual/Logical SymetricSwap=On/Off NumericSwap=On/Off	<p>Utilice la palabra clave BIDIPROP para establecer los valores predeterminados de idioma bidireccional en lenguajes SCLM. LANG= se puede establecer tanto para todos los lenguajes SCLM como para lenguajes SCLM específicos. El soporte bidireccional sólo se soporta en Developer for System z.</p>
PROJECTLISTALL=Y	<p>Si se establece este distintivo de lista de proyectos en N, los usuarios no podrán seleccionar * como filtro de proyecto. Es posible que esto evite prolongadas búsquedas de catálogo de usuario para todos los proyectos SCLM.</p>

Ejemplo de utilización de combinaciones de alteraciones temporales de TRANSLATE.conf

El archivo TRANSLATE.conf configura valores predeterminados para el soporte de página de códigos y para el soporte de lenguaje SCLM predeterminado que deberán aplicarse por SCLM Developer Toolkit. En este ejemplo, TRANSLATE.conf tiene los valores listados a continuación.

```

CODEPAGE ASCII = ISO8859-1
CODEPAGE EBCDIC = IBM-1047
*
TRANLANG JAVABIN
TRANLANG J2EEBIN
TRANLANG J2EEOBJ
TRANLANG TEXTBIN
TRANLANG BINARY
TRANLANG DOC
TRANLANG XLS
*
LONGLANG JAVA
LONGLANG SQLJ
LONGLANG DOC
LONGLANG XLS
LONGLANG J2EEPART
LONGLANG JAVABIN
LONGLANG J2EEBIN
LONGLANG J2EEOBJ

```

Se pueden alterar temporalmente los valores predeterminados de aquellos proyectos diferentes SCLM que almacenen distintos tipos de datos, quizás en diferentes idiomas nacionales. De esta manera, el archivo de configuración SCLMPRJ1.conf para SCLM, proyecto SCLMPRJ1, puede tener los siguientes valores de alteración temporal:

```

* Alteraciones temporales de página de códigos árabe
*
ASCII=UTF-8
EBCDIC=IBM-420
*
* Entradas TRANLANG y LONGLANG específicas de proyecto
*
TRANLANG=DOC
LONGLANG=DOC

```

Establece páginas de códigos para conversiones fuente en la pareja de páginas de códigos árabes. De forma adicional, se añadirá un lenguaje SCLM de DOC a los valores predeterminados de TRANSLATE.conf.

Es posible que el proyecto SCLMPRJ2 de SCLM tenga distintos valores de alteración temporal en SCLMPRJ2.conf:

```
* Alteraciones temporales de la página de códigos hebrea
*
ASCII=UTF-8
EBCDIC=IBM-424
*
* Entradas TRANLANG y LONGLANG específicas de proyecto
*
TRANLANG=DOC
TRANLANG=XLS
NOTRANLANG=JAVABIN
NOTRANLANG=J2EEBIN
NOTRANLANG=J2EEOBJ
LONGLANG=DOC
LONGLANG=XLS
NOLONGLANG=COBOL
NOLONGLANG=J2EEPART
NOLONGLANG=JAVABIN
NOLONGLANG=J2EEBIN
NOLONGLANG=J2EEOBJ
```

Establece páginas de códigos para conversiones fuente en la pareja de páginas de códigos hebreas. De forma adicional, se añadirán lenguajes SCLM de DOC y XLS a los valores predeterminados de TRANSLATE.conf. En tal caso, los valores establecidos en TRANSLATE.conf se eliminarán. Este paso no es realmente necesario, puesto que tener valores adicionales no es un problema, pero demuestra cómo se puede configurar un proyecto para que sólo tenga los lenguajes SCLM necesarios para un proyecto SCLM específico.

Ejemplo de utilización de combinaciones de alteraciones temporales de BIDIPROP

Los valores BIDIPROP especificados en SITE.conf pueden ser alterados temporalmente por cualquiera de los valores BIDIPROP especificados en los archivos <project>.conf específicos de proyecto SCLM. Por ejemplo, se ha establecido lo siguiente en SITE.conf:

```
*
* ----- OPCIONES BIDI ESPECÍFICAS DE SITE -----
*
*
* Propiedades predeterminadas de lenguaje bidireccional
*
BIDIPROP=LANG=*      TextOrient=LTR TextType=Visual SymetricSwap=Off NumericSwap=Off
```

Establece todos los lenguajes SCLM en los valores especificados. Ahora se puede establecer lo siguiente en el archivo ADMIN10.conf:

```
*
* Propiedades predeterminadas de lenguaje bidireccional
BIDIPROP=LANG=JAVA TextOrient=RTL TextType=Visual SymetricSwap=On NumericSwap=Off
BIDIPROP=LANG=COBOL TextOrient=RTL TextType=Logical SymetricSwap=Off NumericSwap=Off
```

Estos valores alterarán temporalmente los valores de SITE.conf para las definiciones de lenguaje de JAVA y COBOL. El resto de lenguajes tienen los valores predeterminados especificados en SITE.conf.

Capítulo 3. Soporte SQLJ

SQLJ es una ampliación de lenguaje para Java. Es una de las muchas tecnologías que permiten a los programadores Java incluir la comunicación de base de datos en sus programas. SQLJ proporciona un método para producir SQL integrado y estático que normalmente genera equivalentes dinámicos tales como JDBC.

CLM Developer Toolkit se suministra con scripts de muestra que permiten construir programas Java habilitados para SQLJ utilizando DB2.

La finalidad de la lectura de este capítulo es que entienda lo esencial de SQLJ y cómo aplicar este conocimiento al utilizar SCLM Developer Toolkit.

¿Qué es SQL?

SQL es un acrónimo de *Structured Query Language* (lenguaje de consulta estructurado). Se trata de un lenguaje abierto, utilizado para consultar, añadir, eliminar y modificar datos en un sistema de gestión de bases de datos relacionales (RDMS).

La primera implementación de este lenguaje fue en un antiguo producto de base de datos de IBM en los años 70: System R. Desde entonces, SQL ha crecido, se ha estandarizado (por ANSI e ISO) y ha aparecido bajo diferentes formas en muchos sistemas de base de datos distintos.

¿Qué es DB2?

DB2 es un conocido sistema de bases de datos, tradicionalmente para la plataforma de sistema principal, que desde su inicio se ha ampliado a muchos otros. Es el estándar para los sistemas de gestión de bases de datos relacionales en z/OS.

DB2 UDB Versión 8 es la versión en la que se basan los scripts de construcción de SCLM Developer Toolkit. Las referencias a DB2 de este capítulo se refieren específicamente a DB2 UDB Versión 8.

¿Qué es JDBC?

JDBC significa *Java Database Connectivity* (conectividad de bases de datos Java). En desarrollo de Java se trata de una tecnología muy conocida que se utiliza de forma habitual para implementar la interacción de bases de datos. JDBC es una API de nivel de llamada, lo que significa que las sentencias SQL se pasan como series a la API que, posteriormente, se encarga de ejecutarlas en RDMS. Por ello, el valor de estas series se puede modificar durante el tiempo de ejecución, haciendo que JDBC sea dinámica.

Mientras que los programas JDBC se ejecutan de forma más lenta que sus equivalentes SQLJ, una ventaja de este método es un concepto denominado "Write once, call anywhere" (escribir el código una sola vez y ejecutarlo en cualquier plataforma). Esto significa que, puesto que no se necesita ninguna interacción hasta el tiempo de ejecución, un programa JDBC es muy portable y se puede emplear entre dos sistemas distintos sin ningún tipo de preocupación.

¿Qué es SQLJ?

SQLJ es una ampliación de lenguaje utilizada para transacciones de base de datos en aplicaciones Java. Genera SQLJ integrado y estático. Este término está formado por SQL, que quiere decir *Structured Query Language* y J, que significa *Java*.

SQLJ es *estático* porque las sentencias SQL que se ejecutarán durante el tiempo de ejecución se conocen cuando se ensambla el programa. Al contrario que JDBC, donde las consultas que se ejecutan se pueden cambiar en cualquier momento.

SQLJ está *incorporado* porque durante el establecimiento de enlaces se ofrece a la base de datos un formato serializado de sentencias SQL de los programas a la base de datos. La base de datos utiliza estos datos serializados para determinar vías de acceso optimizadas a las tablas a las que hacen referencia. En JDBC, la base de datos no puede determinar qué sentencias se ejecutarán hasta que las recibe durante el tiempo de ejecución desde la aplicación. Por ello, debe determinar vías de acceso durante el tiempo de ejecución. Esto incurre en una sobrecarga que se puede evitar utilizando SQLJ.

Comparación entre JDBC y SQLJ

Esta tabla se basa en material localizado en la sección 5.2 del Redbook *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*.

Tabla 7. Comparación entre JDBC y SQLJ

	SQLJ (estático)	JDBC (dinámico)
RENDIMIENTO	La mayoría de las veces, el SQL estático es más rápido que el SQL dinámico, porque durante el tiempo de ejecución sólo debe comprobarse la autorización de paquetes y de planes antes de ejecutar el programa.	Las sentencias de SQL dinámico necesitan que se analicen las sentencias SQL, que se compruebe la autorización de tabla/vista y que se determine la vía de acceso de optimización.
AUTORIZACIÓN	Con SQLJ, el propietario de la aplicación otorga la autoridad EXECUTE al plan o al paquete y el destinatario del GRANT debe ejecutar la aplicación tal como se ha escrito.	Con JDBC, el propietario de la aplicación otorga privilegios en todas las tablas subyacentes utilizadas por la aplicación. El destinatario de dichos privilegios puede hacer todo lo que le permitan los mismos, como por ejemplo utilizarlos fuera de la aplicación para las que se otorgaron las autorizaciones originariamente. La aplicación no puede controlar lo que puede hacer el usuario.
DEPURACIÓN	SQLJ no es una API sino una ampliación del lenguaje. Esto significa que las herramientas de SQLJ tienen conocimiento de las sentencias SQL del programa y comprueban su sintaxis y sus autorizaciones durante el proceso de desarrollo del programa.	JDBC es una API de nivel de llamada pura. Esto significa que el compilador Java no tiene conocimiento de las sentencias SQL; sólo aparecen como argumentos de las llamadas de método. Si una de las sentencias tiene un error, no podrá detectarlo hasta el tiempo de ejecución, cuando la base de datos se "queje" del mismo.

Tabla 7. Comparación entre JDBC y SQLJ (continuación)

	SQLJ (estático)	JDBC (dinámico)
SUPERVISIÓN	Con SQLJ se puede mejorar la supervisión del sistema y la presentación de informes de rendimiento. Los paquetes de SQL estático indican los nombres de los programas que se están ejecutando en un momento dado. Esto resulta especialmente útil para estudiar el consumo de CPU de las distintas aplicaciones, los problemas de bloqueo (tales como puntos muertos o tiempos de espera excedidos), etc.	Mientras que en SQLJ se puede determinar el nombre del programa que está ejecutando actualmente, con JDBC todas las transacciones tienen lugar a través del mismo programa. Esto hace que la supervisión y la ubicación de las áreas problemáticas sean más difíciles.
VERBOSIDAD	Puesto que las sentencias SQL están codificadas con sintaxis SQL pura sin tener que derivarlas en un método de Java, los propios programas resultan más fáciles de leer, lo que también facilita su mantenimiento. Además, puesto que algunos de los códigos modelo que se deben que codificar explícitamente en JDBC se generan automáticamente en SQLJ, los programas escritos en SQLJ tienden a ser más breves que los programas JDBC equivalentes.	Con JDBC, todas las sentencias SQL deben derivarse en llamadas de API que generalmente producen código de poca claridad y concisión.

¿Qué es un perfil serializado?

Un Perfil serializado es un código que se graba en SQLJ y se coloca en un archivo con una extensión `.sqlj`. En el primer paso de la preparación del programa (posteriormente se abordará de forma más detallada), el archivo `.sqlj` se introduce en el conversor SQLJ.

El conversor genera dos tipos de salida. La primera es código fuente Java (`.java`). El código fuente es, obviamente, la implementación Java del código contenido en el archivo `.sqlj`.

El segundo tipo de salida es un perfil serializado (`.ser`). Este archivo contiene todas las sentencias SQL del archivo `.sqlj` en un formato serializado. Este perfil debe estar disponible para el programa durante el tiempo de ejecución y también se debe poder utilizar para enlazarlo con RDMS.

¿Qué es un DBRM?

DBRM significa *Database Request Module* (Módulo de petición de base de datos). Se trata de la representación serializada de DB2 tradicional de las sentencias SQL de un programa. Por ejemplo: un programa se puede escribir en COBOL. DB2 preprocesará este programa para producir un DBRM que se utilizará para establecer un enlace con un subsistema DB2 en particular.

Con SQLJ el proceso es ligeramente distinto y, en DB2 UDB Versión 8, se hace referencia al mismo como *modalidad de compatibilidad*. El programa de utilidad `db2sqljcustomize` se puede suministrar con argumentos de línea de mandatos opcionales que hacen que se genere un DBRM. A continuación, este DBRM se

puede enlazar con DB2 utilizando los medios tradicionales. Por ejemplo, un script REXX llamado por una salida de usuario SCLM.

Preparación del programa SQLJ

Antes de abordar cómo utilizar SCLM Developer Toolkit para construir programas SQLJ, examinaremos primero el proceso manual. Este proceso está previsto para la implementación DB2 de SQLJ e incluye 3 mandatos denominados *sqlj*, *db2sqljcustomize* y *db2bind*. Tenga en cuenta que el paso de enlace se puede realizar opcionalmente en *db2sqljcustomize*, por lo que no siempre se necesita *db2bind*.

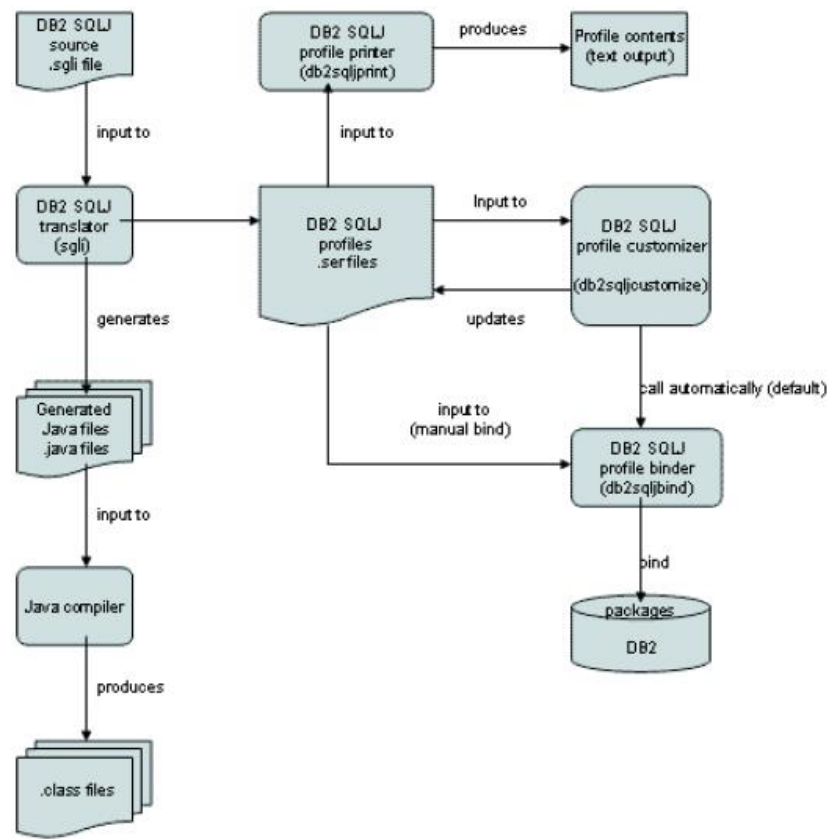


Figura 17. Preparación del programa SQLJ

Conversión

El conversor SQLJ (que no debe confundirse con un *conversor de lenguaje SCLM*) toma los archivos fuente SQLJ como entrada y produce código fuente de Java (archivos .java) y perfiles serializados (archivos .ser).

El lenguaje SQLJ en sí no se aborda en este manual. Consulte <http://www.sql.org> para obtener referencias sobre el desarrollo de código SQLJ.

El número de perfiles serializados generado por cada archivo .sqlj depende del número de clases de *contexto de conexión* referenciadas dentro del código SQLJ. Se generará un perfil serializado para cada uno.

Muchos archivos fuente SQLJ sólo harán referencia a una única clase de contexto de conexión y, por ello, sólo generarán un único perfil serializado. Los perfiles

serializados se nombran siguiendo el orden en que se referencian en el archivo fuente. El nombre adopta el formato siguiente:

nombre_prog_SJProfileX.ser

Donde:

- *nombre_prog* representa el nombre del programa. Se determina eliminando la extensión .sqlj del nombre de archivo de la fuente de entrada.
- *X* representa un número entero que simboliza el índice de la clase actual. La indexación se basa en el cero. La primera clase de contexto de conexión referenciada producirá el perfil 0; la segunda producirá el perfil 1 y así sucesivamente.

Ejemplo:

Entrada: Customer.sqlj (hace referencia a una clase de contexto de conexión)

Salida: Customer.java
Customer_SJProfile0.ser

Y, opcionalmente, si se suministra el argumento -compile=true a sqlj:
Customer.class

Personalización

Una vez generados los perfiles serializados, los personalizamos. El mandato para hacerlo en DB2 Versión 8 es *db2sqljcustomize*; no obstante, en versiones anteriores era *db2prof*. Cada invocación del personalizador debería coincidir con una invocación del conversor de SQLJ. En otras palabras, si una única invocación del conversor ha generado cinco perfiles, dichos cinco perfiles deben introducirse como entrada de una única invocación del personalizador de perfiles. Otra forma de imaginarlo es asociar cada nombre de programa individual con una invocación de cada uno de los programas de utilidad. Recuerde que el nombre del programa es el mismo que el del nombre de archivo de la fuente de entrada con la extensión .sqlj eliminada.

La personalización añade información específica de DB2 al perfil serializado que se utiliza durante el tiempo de ejecución. Otras opciones, tales como el enlace automático, se pueden configurar mediante los conmutadores de línea de mandatos. Si está utilizando una versión preexistente de DB2 o si especifica los distintivos *gendbrm* y *dbrmdir* para *db2sqljcustomize*, se generará un archivo DBRM. Este archivo se utiliza posteriormente para enlazarlo con la base de datos. Con el controlador universal de DB2 UDB 8+, puede renunciar a la generación de un DBRM y, en su lugar, realizar el enlace utilizando los perfiles serializados generados por el conversor SQLJ.

Establecimiento de enlaces

El enlace es el último paso del proceso de preparación del programa SQLJ. En DB2 Versión 8, el mandato utilizado para enlazar es *db2sqljbind* pero también puede establecer el enlace automáticamente ejecutando *db2sqljcustomize*. El enlace es el paso que construye una vía de acceso a las tablas DB2 para las sentencias SQL serializadas. Estas sentencias están disponibles en forma de DBRM o de perfil serializado.

De forma predeterminada, se crean cuatro paquetes, uno para cada nivel de aislamiento. Puede establecer el enlace utilizando el método tradicional, donde se utiliza un DBRM, o utilizando el nuevo Método universal, donde se utilizan perfiles serializados en su lugar.

Tipos y conversores de SCLM DT

Antes de abordar los tipos y conversores de SCLM, debe realizarse una importante distinción entre un *Conversor de lenguaje SCLM* o, simplemente, *Conversor SCLM*, y el conversor SQLJ *sqlj* que se suministra como parte de DB2.

En SCLM todos los lenguajes definidos deben tener un conversor para saber cómo tratar dicho lenguaje. No ocurre lo mismo con el conversor SQLJ, *sqlj* que es un programa de utilidad de línea de mandatos que toma un archivo fuente SQLJ y produce perfiles serializados y código fuente Java.

Una vez realizada tal distinción, abordaremos los *Tipos SCLM* y los *Conversores SCLM* asociados al proceso de construcción SQLJ tal como sigue.

Se suministra un conversor SCLM para SQLJ que debería asignarse como el tipo de lenguaje de todo el código fuente SQLJ almacenado en SCLM. Este nuevo conversor requiere que se definan Tipos SCLM adicionales. El conversor SCLM para SQLJ es parecido al conversor JAVA pero contiene definiciones IOTYPE adicionales para los tipos de salida SQLJSER y DBRMLIB de SCLM. Si no desea generar archivos DBRM como parte del paso de personalización, puede eliminar esta IOTYPE DBRMLIB de la definición de lenguaje SQLJ.

Dentro de la definición de proyecto, un administrador debe definir y generar el nuevo conversor SCLM y los tipos adicionales.

Tabla 8. Tipos de conversores SCLM para SQLJ

SQLJSER	Necesario para almacenar los archivos de perfil serializados generados (archivos .ser) creados o personalizados en los pasos de conversión y de personalización. Es recomendable definir este conjunto de datos de tipo SCLM como recfm=VB, lrecl=256.
DBRMLIB	Necesario para contener los archivos DBRM generados que se han creado en el paso de personalización. Este tipo sólo resulta necesario para los clientes que utilicen archivos DBRM generados como parte de su proceso de enlace DB2.

Ajuste del proceso de construcción

A fin de conservar la máxima flexibilidad, el proceso de construcción SQLJ es altamente personalizable, para tener en cuenta distintas configuraciones de sitio y cualquier combinación de parámetros que deba pasarse a *sqlj* y a *db2sqljcustomize*.

Esta sección describe los conceptos que alberga la implementación de SQLJ de SCLM Developer Toolkit. Tras su lectura, podrá personalizar el proceso de construcción de forma que cumpla con los requisitos de su sitio.

Al efectuar la personalización de perfil y la conversión SQLJ, SCLM Developer Toolkit invoca directamente a las mismas clases Java que utilizan los mandatos *sqlj* y *db2sqljcustomize*. Tenga en cuenta que los argumentos suministrados para los procesos de conversión y de personalización de SCLM DT serán exactamente los

mismos. Para obtener más información sobre todos los argumentos de línea de mandatos de cada mandato, consulte la *Guía del usuario de DB2 Universal Database*.

Ajuste del script de construcción

Si suponemos que ha utilizado el asistente para Añadir a SCLM, al script de construcción del programa SQLJ se le dará el mismo nombre de miembro que la Archdef. Por ejemplo, si la Archdef del proyecto sqlj es SCLM10.DEV1.ARCHDEF(SQLJ01), deberá colocar el script de construcción en SCLM10.DEV1.J2EEBLD(SQLJ01).

En dicho script de construcción habrá una referencia al script de construcción maestro. Éste se puede encontrar en la propiedad. La mayor parte de la configuración listada para estos pasos de conversión y de personalización sigue en este archivo.

Nota: El script de construcción suministrado con Developer Toolkit es BWBSQLB para los proyectos JAR y BWBSQLBE para los proyectos EJB. No debería ser necesario cambiar este valor.

Propiedades de sqlj.*

Cada propiedad listada en la Tabla 9 aparece en el script de construcción BWBSQLB. Las propiedades están en formato XML de la siguiente manera:

La configuración del script implica cambiar el valor de todas las propiedades relevantes.

Tabla 9. Propiedades de sqlj.*

Nombre	Valor	Descripción
sqlj.exec	usr/lpp/rdz/bin/bwbsqlc.rex	Especifica la ubicación de la rutina exec sqlj & db2sqljcustomize bwbsqlc.rex, que está ubicada en el directorio de instalación de Developer for System z.
sqlj.class	sqlj.tools.Sqlj	Especifica el nombre de clase sqlj. Se trata del nombre de la clase invocada por el programa de utilidad sqlj. Es muy poco probable que deba cambiar este valor.
sqlj.bin	/db2path/bin	Especifica la ubicación del directorio bin de sqlj de db2 donde reside el script sqlj.
sqlj.cp	/db2path/jcc/classes/sqlj.zip	Especifica la ubicación de sqlj.zip para la inclusión en la vía de acceso.
sqlj.arg	-compile=false status linemap=NO db2optimize	Especifica los argumentos de propiedad globales que aparecen a continuación para el proceso sqlj.

Propiedades de db2sqljcustomize.*

Cada propiedad listada en la Tabla 10 en la página 42 aparece en el script de construcción BWBSQLB. Las propiedades están en formato XML de la siguiente manera:

```
<property name= NAME value= VALUE />
```

La configuración del script implica cambiar el valor de todas las propiedades relevantes.

Tabla 10. propiedades de db2sqljcustomize.*

Nombre	Valor	Descripción
sqljdb2cust.class	com.ibm.db2.jcc.sqlj.Customizer	Especifica el nombre de clase de personalización de DB2 sqlj. Es muy poco probable que deba modificar este valor.
db2sqljcust.cp	/db2path/jcc/classes/db2jcc.jar: ./SRC: /db2path/jcc/classes/db2jcc_license_cisuz.jar	Valores de vía de acceso para el programa de utilidad de personalización. Deben suministrarse nombres de vía de acceso calificados al completo en XML.
db2sqljcust.arg	-automaticbind NO -onlinecheck YES -staticpositioned YES -bindoptions â ISOLATION(CS)â -genDBRM	Argumentos generales que deben suministrarse para el programa de utilidad de personalización.
db2sqljcust.propfile	user.properties	Nombre de archivo de propiedades que debe pasarse a un script de determinación de propiedades de usuario para valores de propiedad dinámicos. Déjelo de forma predeterminada.
db2sqljcust.userpgm	NONE si desea evitar el script. De lo contrario, especifique la vía de acceso calificada al completo y el nombre de archivo del script de usuario.	Este script se ejecutará inmediatamente antes del programa de utilidad de personalización. Actualiza dinámicamente un archivo de propiedades que se utiliza como entrada para el programa de utilidad de personalización.

Script de usuario personalizado

El script de construcción SQLJ suministrado por SCLM Developer Toolkit está diseñado para funcionar en modalidad de compatibilidad DB2 UDB v8. Esta modalidad soporta el concepto DB2 de DBRM, en lugar de establecer enlaces mediante los perfiles serializados. Para utilizar los perfiles serializados deben realizarse cambios en BWBSQLB. Este tema se aborda en el subtema “Establecimiento de enlaces [Perfil serializado]” en la página 45.

Además de la metodología de enlace, para que haya correspondencia entre el proceso de construcción y su sitio, es probable que deba personalizar los argumentos de *sqlj* y *db2sqljcustomize* para que coincidan con su entorno de base de datos, su política de aislamiento y otros factores. Es posible incluso que desee establecer sus propios scripts para determinar propiedades dinámicas para estos argumentos. Por ejemplo, si desea crear de forma inteligente un nombre de paquete relacionado con el nombre del archivo de entrada.

SCLM Developer Toolkit permite hacerlo especificando su propio script de personalización. Todo lo que existe en el proceso de construcción XML de Ant funciona en el concepto de “propiedades”, elementos de la *Propiedad* XML que especifican una pareja de nombre o valor. Por ejemplo, en el paso *db2sqljcustomize* del script de construcción BWBSQLB, los argumentos de línea de mandatos globales que deben suministrarse para *db2sqljcustomize* se definen en un elemento de

propiedad con el nombre *db2sqljcust.arg* y un valor predeterminado de *-automaticbind NO -onlinecheck YES -staticpositioned YES -bindoptions "ISOLATION(CS)" genDBRM lang=EN-AU*.

Si desea modificar los argumentos suministrados, puede editar el script de construcción para cambiar el valor de la propiedad, cambiando los valores de forma global, o puede ensartar su propio script de personalización en el proceso.

Para ensartar su propio script de propiedades personalizado, ponga el nombre del script en *db2sqljcust.userpgm* y el nombre del archivo de propiedades que desee grabar en *db2sqljcust.propfile*.

El script especificado en *db2sqljcust.userpgm* se ejecutará justo antes que el proceso *db2sqljcustomize*. Su script actualizará automáticamente el archivo de propiedades especificado en *db2sqljcust.userpgm*. Este archivo de propiedades se utilizará como entrada para el proceso *db2sqljcustomize*, porque el proceso de construcción concatena las dos propiedades en el archivo de propiedades actualizado de forma dinámica y las propiedades ya definidas en el script de construcción.

Los siguientes argumentos le suministrarán el script especificado en *db2sqljcust.userpgm* cuando lo ejecute:

Argumento	Descripción
Basedir	Directorio base (directorio de espacio de trabajo)
Propfile	El nombre del archivo de propiedades que se debe crear y actualizar Nota: El archivo de propiedades que se está creando debe ser <i>basedir'/'propfile</i> .
Sqljf	Una lista de nombres de archivo que representan los perfiles serializados (.ser) que <i>db2sqljcustomize</i> debe procesar

Las propiedades deberían establecerse en el archivo con el siguiente formato, con una declaración de propiedad por línea:

```
argument=value
Por ejemplo,
singlepkgname= pkgname
```

Por ejemplo:

```
pkgversion=1
url=jdbc:db2://site1.com:80/MVS01
qualifier=DBT
singlepkgname= SQLJ986
```

Se llama a la rutina personalizada una vez por archivo. Finalmente, las propiedades de argumento se utilizan para construir la serie de argumento para la llamada *db2sqljcustomize*. Por ejemplo:

```
db2sqljcustomize -automaticbind NO -collection ${db2.collid}
-url ${db2.url} -user ${db2.user} -password ??????? -onlinecheck YES
-qualifier ${db2.qual} -staticpositioned YES -pkgversion ${db2.packversion}
-bindoptions "ISOLATION (CS)"
-genDBRM -DBRMDir DBRMLIB
-singlepkgname ${db2.pack}
```

Establecimiento de enlaces [DBRM]

El DB2 tradicional utiliza un *Módulo de petición de base de datos* o DBRM para este fin. El mandato *db2sqljcustomize* genera el DBRM cuando se suministra el distintivo *gendbrm*. Sin este distintivo, el mandato asumirá que desea establecer los enlaces mediante perfiles serializados y no generará ningún DBRM.

Si suministra este parámetro, SCLM Developer Toolkit identificará los DBRM generados y los almacenará en SCLM para usos posteriores. Una ventaja de la utilización de esta técnica es que permite establecer fácilmente un enlace DB2 en una salida de usuario SCLM, como la salida de construcción/copia.

Puesto que la salida de usuario de construcción/copia se suministra automáticamente con una lista de objetos actualizados, es posible volver a enlazar selectivamente sólo aquellos módulos que se hayan cambiado, evitando la pérdida de eficacia debido a enlaces redundantes.

Existen los cuatro pasos siguientes para configurar enlaces para DBRM:

1. Establezca los argumentos adecuados para *sqlj* de la siguiente manera:

```
<!-- especifique argumentos de propiedad globales debajo para proceso sqlj -->
<property name="sqlj.arg"
value="-compile=false -status -linemap=no"/>
```

Argumento	Descripción
compile=false	Si establece esta opción en FALSE (falsa), evitará que el conversor sqlj compile automáticamente la fuente Java que produce. SCLM Developer Toolkit utiliza la fuente generada en su propio proceso de construcción, por lo que es recomendable que establezca siempre esta opción en FALSE.
linemap=no	Especifica si los números de línea de las excepciones Java coincide con los números de línea de los archivos fuente SQLJ (el archivo .sqlj) o con los números de línea del archivo fuente Java generado por los archivos de conversor de SQLJ (el archivo .java). Se necesita un archivo .class, por lo que debe establecer en <i>no</i> cuando se utiliza en conjunción con <i>compile=false</i> .
status	Imprime la visualización de estado inmediata del proceso SQLJ.

2. Establezca los argumentos adecuados para *db2sqljcustomize*, incluido *gendbrm* de la siguiente manera:

```
<property name="db2sqljcust.arg"
Value='-automaticbind NO -onlinecheck YES
-bindoptions "ISOLATION(CS)" -gendbrm' />
```

Argumento	Descripción
automaticbind no	Si se establece en "no", el personalizador no establecerá un enlace cuando finalice la personalización.
onlinecheck yes	Realiza la comprobación en línea en el sistema especificado por el parámetro <i>url</i> . El valor predeterminado es "yes" si se suministra <i>url</i> y "no" en caso contrario.
Bindoptions ISOLATION(CS)	Indica al enlazador que debe crear un único paquete (estabilidad de cursor). Se utiliza en conjunción con <i>singlepkgname</i> (establecido dinámicamente).
gendbrm	Indica al personalizador que debe generar archivos DBRM.

3. Configure el script de usuario.

Establezca la ubicación del programa de usuario en BWBSQLB. Éste le dice al proceso de construcción dónde puede encontrar el script rex utilizado para calcular propiedades dinámicas.

La propiedad grande que queremos configurar dinámicamente es *singlepkgname*. Se trata del nombre del paquete con el que queremos enlazar, y cada programa va a tener su propio nombre de paquete exclusivo, que en este sencillo ejemplo serán las ocho primeras letras del nombre del programa.

4. Escriba una salida de construcción para enlazar el DBRM. Se recomienda la salida de copia de construcción.

Puesto que está utilizando *singlepkgname* en el paso de personalización, el nombre del paquete será el mismo que el nombre del DBRM.

Establecimiento de enlaces [Perfil serializado]

El nuevo método recomendado para enlazar programas SQLJ consiste en utilizar los perfiles serializados (archivos .ser) para enlazar. Era inevitable, puesto que el perfil serializado realiza la misma función que el DBRM que es suministrar una imagen serializada de las sentencias de dentro del programa.

Con algunas pequeñas modificaciones en el script de construcción BWBSQLB, puede configurar SCLM Developer Toolkit para que utilice este método en su lugar. Simplemente se trata de cambiar los argumentos suministrados a *db2sqljcustomize* para eliminar el conmutador de la línea de mandatos *gendbrm* y cambiar *automaticbind* a YES.

Para configurar el establecimiento de enlaces para perfiles serializados, existen los tres pasos siguientes:

1. Establezca los argumentos adecuados para *sqlj* de la siguiente manera:

No existen argumentos de línea de mandatos para el conversor de *sqlj* que sean exclusivos para el establecimiento de enlaces de perfiles serializados. No obstante, a continuación se muestra el conjunto de argumentos para este ejemplo en particular.

```
<!-- especifique argumentos de propiedad globales debajo para proceso sqlj -->
<property name="sqlj.arg"
value="-compile=false -status -linemap=no"/>
```

Argumento	Descripción
compile=false	Si establece esta opción en FALSE (falsa), evitará que el conversor <i>sqlj</i> compile automáticamente la fuente Java que produce. SCLM Developer Toolkit utiliza la fuente generada en su propio proceso de construcción, por lo que es recomendable que establezca siempre esta opción en FALSE.
linemap=no	Especifica si los números de línea de las excepciones Java coincide con los números de línea de los archivos fuente SQLJ (el archivo .sqlj) o con los números de línea del archivo fuente Java generado por los archivos de conversor de SQLJ (el archivo .java). Se necesita un archivo .class, por lo que debe establecer en <i>no</i> cuando se utiliza en conjunción con <i>compile=false</i> .
status	Imprime la visualización de estado inmediata del proceso SQLJ.

2. Establezca los argumentos adecuados para *db2sqljcustomize* de la siguiente manera:

```
<property name="db2sqljcust.arg"
Value='-automaticbind YES -onlinecheck YES'/>
```

Argumento	Descripción
automaticbind yes	Si se establece en "yes", el personalizador también establecerá un enlace cuando finalice la personalización. Si se establece en "no", el enlace deberá realizarse por separado con el mandato <i>db2bind</i> .
onlinecheck yes	Realiza la comprobación en línea en el sistema especificado por el parámetro <i>url</i> . El valor predeterminado es "yes" si se suministra <i>url</i> y "no" en caso contrario.

3. Configure el script de usuario.

Establezca la ubicación del programa de usuario en BWBSQLB. Éste le dice al proceso de construcción dónde puede encontrar el script *rex* utilizado para calcular propiedades dinámicas.


```
<property name="db2sqljcust.userpgm" value="/u/dba/sqljcust.rex"/>
```

Capítulo 4. Seguridad de SCLM

SCLM Developer Toolkit ofrece una funcionalidad de seguridad opcional para las funciones de Construcción, Promoción y Despliegue.

- La funcionalidad de seguridad está controlada por distintivos de seguridad establecidos en los archivos de configuración de SCLM Developer Toolkit y por perfiles en el producto de seguridad.
- Si se establece el distintivo de seguridad para una función y el ID de usuario que hace la solicitud cumple con la comprobación de autoridad efectuada por el producto de seguridad, se permite que prosiga la función. Si el solicitante no pasa la comprobación de seguridad, la función finaliza con RC=8 y un mensaje de error de seguridad.
- Si así se ha definido en el producto de seguridad, la función proseguirá utilizando otro ID de usuario (sustituto).

Distintivo de seguridad

Puede establecer un distintivo de seguridad de construcción, promoción o despliegue en los archivos de configuración SITE/PROJECT. Consulte “SITE y opciones específicas de proyecto” en la página 27 para obtener más información acerca de los archivos de configuración SITE/PROJECT. Las directivas siguientes controlan el distintivo de seguridad de las funciones de construcción, promoción y despliegue, respectivamente:

- BUILDSECURITY = Y
- PROMOTESECURITY = Y
- DEPLOYSECURITY = Y

Configuración del producto de seguridad

La seguridad de la función de construcción, promoción y despliegue de SCLM utiliza la interfaz de seguridad SAF/RACROUTE, a la que dan soporte todos los productos de seguridad principales.

Los mandatos de muestra listados están previstos para RACF. Consulte la documentación del producto de seguridad si utiliza otro producto.

- Utilice el mandato RDEFINE para definir en RACF todos los recursos que pertenezcan a las clases especificadas en la tabla de descriptor de clases. El mandato RDEFINE añade un perfil para el recurso en la base de datos RACF a fin de controlar el acceso al recurso.
- El mandato PERMIT se utiliza para mantener las listas de usuarios y grupos autorizados para acceder a un recurso en particular. La lista de accesos estándar incluye los ID de usuario y los nombres de grupo que disponen de autorización para acceder al recurso y al nivel de acceso otorgado a cada uno.
- Defina perfiles de función para las funciones de SCLM en la clase XFACILIT.
- El administrador de seguridad define los perfiles RACF necesarios utilizando el mandato RDEFINE y le permite acceder con el mandato PERMIT.

Perfiles de seguridad

El administrador de seguridad define los perfiles necesarios en la clase XFACILIT utilizando el mandato **RDEFINE** y los permisos de acceso con el mandato **PERMIT**. Consulte la Tabla 11 para obtener más información sobre los perfiles que deben definirse.

Tabla 11. Perfiles de seguridad de SCLMDT Developer Toolkit

Función	Perfil XFACILIT	Acceso necesario
Construcción	SCLM.BUILD.project.projdef.group.type.member	READ
Promoción	SCLM.PROMOTE.project.projdef.group.type.member	READ
Despliegue	SCLM.DEPLOY.server.application.node.cell.project.projdef.group.type	READ

La lista que encontrará a continuación describe el significado de los distintos nombres de partes de perfil:

SCLM	Constante, indica un perfil de función SCLM
BUILD	Constante, indica la función BUILD
PROMOTE	Constante, indica la función PROMOTE
DEPLOY	Constante, indica la función DEPLOY
project	El nombre del proyecto SCLM o * para todos los proyectos
projdef	El nombre del proyecto alternativo (equivale al nombre del proyecto predeterminado) o * para todos los proyectos alternativos
Group	El grupo SCLM que se debe construir/promover/desplegar o * para todos los grupos
Type	El tipo SCLM o * para todos los tipos
Member	El miembro SCLM que se debe construir/promover o * para todos los miembros
Server	El servidor de despliegue de destino (SERVER_NAME en el script de despliegue Ant) o * para todos los servidores
Application	El nombre de la aplicación WAS de destino (APPLICATION_NAME en el script de despliegue de Ant) o * para todas las aplicaciones
Node	El nombre del nodo WAS de destino (NODE_NAME en el script de despliegue de Ant) o * para todos los nodos
Cell	El nombre de la celda WAS de destino (CELL_NAME en el script de despliegue de Ant) o * para todas las celdas

ID de usuario sustituto

Las funciones de construcción, promoción y despliegue soportan el uso de un ID de usuario sustituto para ejecutar la función. Si está activado, todas las llamadas autorizadas a la función causarán que la función se ejecute con los permisos del ID de usuario sustituto, no con el ID de usuario que hace la solicitud.

La activación del ID de usuario sustituto es específico de perfil y la controla la serie "SUID=userid" en el campo APPLDATA del perfil de seguridad, donde IDusuario es el ID de usuario sustituto. Si la serie está presente se utilizará el ID de usuario sustituto y, si no lo está, se utilizará el ID de usuario solicitante.

Ejemplo: construcción

Este ejemplo lista las definiciones de producto de seguridad necesarias para proteger la función de construcción para un proyecto determinado. Se puede

utilizar el mismo ejemplo para proporcionar seguridad a la función de promoción sustituyendo la regla SCLM.BUILD.* por la regla SCLM.PROMOTE.*.

La siguiente definición de perfil proporciona seguridad a todos los miembros para la construcción en project=TESTPROJ en PROD de nivel de grupo. También se define un ID de usuario sustituto.

```
RDEFINE XFACILIT SCLM.BUILD.TESTPROJ.TESTPROJ.PROD.*.* UACC(NONE)
          APPLDATA('SUID=IBMUSER')
SETROPTS RACLIST(XFACILIT) REFRESH
```

Este ejemplo define un perfil de construcción SCLM donde:

- Proyecto = TESTPROJ
- Definición de proyecto alternativa = TESTPROJ
- Grupo SCLM = PROD
- Tipo SCLM = Todos los tipos
- Miembro = Todos los miembros
- Acceso universal = NONE (de forma predeterminada, nadie dispone de autorización para el perfil)
- ID de usuario sustituto = IBMUSER

Nota: El mandato SETROPTS actualiza las tablas en memoria de RACF y, de esta manera, se activa la definición.

El ejemplo siguiente muestra permisos de seguridad definidos para usuarios individuales (o grupos de usuarios) para el proyecto TESTPROJ del ejemplo anterior:

```
PERMIT SCLM.BUILD.TESTPROJ.TESTPROJ.PROD.*.* CLASS(XFACILIT) ID(USERID) ACCESS(READ)
```

PERMIT encuentra el perfil RDEFINE original y permite que el usuario USERID construya cualquier miembro del proyecto TESTPROJ y del grupo PROD. Puesto que existe un ID de usuario sustituto almacenado en el campo de datos de aplicación(APPLDATA) de la regla de coincidencia, BUILD se procesará bajo el ID de usuario sustituto (en este caso, IBMUSER).

Ejemplo: despliegue

A continuación encontrará un ejemplo de creación de un perfil de despliegue.

```
RDEFINE
XFACILIT SCLM.DEPLOY.SERVERY.TESTAPP.NODE1.CELL1.TESTPROJ.TESTPROJ.PROD.J2EEDEP
UACC(NONE)
```

Detalles de WAS:

- Nombre de servidor = SERVERY
- Nombre de aplicación = TESTAPP
- Nombre de nodo = NODE1
- Nombre de celda = CELL1

Detalles del proyecto SCLM:

- Proyecto = TESTPROJ
- Definición de proyecto alternativa = TESTPROJ
- Grupo SCLM = PROD
- Tipo SCLM = J2EEDEP

Otra información:

- Acceso universal = NONE (de forma predeterminada, nadie dispone de autorización para el perfil)
- Ningún ID de usuario sustituto

El siguiente ejemplo muestra permisos de seguridad definidos para un grupo de usuarios para el perfil de despliegue previamente definido:

```
PERMIT SCLM.DEPLOY.SERVERY.TESTAPP.NODE1.CELL1.TESTPROJ.TESTPROJ.PROD.J2EEDEP  
      CLASS(XFACILIT) ID(J2EEGRP) ACCESS(READ)
```

Se encuentra el perfil RDEFINE original y se permite que cualquier usuario que pertenezca al grupo J2EEGRP de RACF se despliegue en el mencionado y desde los mismos detalles de proyecto SCLM.

Capítulo 5. Construcciones y promociones iniciadas por CRON

Aunque la mayoría de las construcciones y promociones se inician mediante el cliente de Developer Toolkit, existe la posibilidad de configurar archivos de configuración de construcción y de promoción dentro del sistema de archivos de z/OS UNIX System Services y de iniciar estas construcciones o promociones mediante el servicio CRON (tiempo) dentro de UNIX System Services.

Utilizando este método no se necesita el cliente de SCLM Developer Toolkit, puesto que los parámetros de construcción y promoción relevantes se leen desde un archivo de configuración de z/OS UNIX System Services y se pasan al componente host de Developer Toolkit para el proceso de SCLM.

A continuación aparece una descripción de las muestras de SCLM Developer Toolkit que proporcionan construcciones y promociones iniciadas por CRON. Estas muestras están disponibles en el directorio de muestra de Developer para System z, /usr/lpp/rdz/samples/. Se ha colocado una copia (que se puede personalizar para que cumpla con sus requisitos) en /etc/rdz/scldmt/script/ durante la personalización de Developer para System z.

BWBCRON1

Esta muestra REXX llama a la interfaz de host de SCLM Developer Toolkit y pasa los parámetros de función. La salida desde el proceso de función se visualiza de forma predeterminada en STDOUT pero se puede redireccionar hacia un archivo o registro de z/OS UNIX. Deberá personalizar REXX tal como se detalla dentro de la muestra. Esta muestra REXX debe ejecutarse en conjunción con la entrada de la muestra BWBCRONB para una construcción o de la muestra BWBCRONP para una promoción.

BWBCRONB

Esta muestra REXX configura la serie de entrada del parámetro de construcción que se pasa al módulo BWBCRON1. La muestra requiere personalizaciones por parte del usuario para actualizar todos los parámetros de construcción necesarios.

BWBCRONP

Esta muestra REXX configura la serie de entrada del parámetro de promoción que se pasará al módulo BWBCRON1. La muestra requiere personalizaciones por parte del usuario para actualizar todos los parámetros de promoción necesarios.

bwbsqld.rex

Script de construcción JAVA SQLJ XML ANT de ejemplo

Requisitos de STEPLIB y PATH

Las variables PATH y STEPLIB del perfil de todo el sistema (/etc/profile) o del perfil del usuario (/u/userid/.profile) deberán establecerse para ubicar los trabajos CRON (\$PATH) y los módulos de carga SCLM Developer Toolkit (\$STEPLIB) si los módulos de carga de SCLM Developer Toolkit no se encuentran en LINKLIST. Por ejemplo:

```
PATH=/etc/rdz/scldmt/script:$PATH
STEPLIB=FEK.SFEKLOAD:FEK.SFEKAUTH:$STEPLIB
```

Ejecución de trabajos de construcción de CRON

Después de añadir trabajos de CRON a la variable PATH, se pueden ejecutar conectando la salida de `parameter_exec` en `processing_exec`. La salida puede entonces dirigirse a un archivo de anotaciones de salida.

Sintaxis

```
parameter_exec | processing_exec > output.log
```

"|" es el símbolo del conducto de z/OS UNIX.

Ejemplo de invocación

Utilizando los nombres de muestra tal como se han suministrado, el `exec` de construcción de CRON se puede invocar de la siguiente manera (\$ es la solicitud de z/OS UNIX):

```
$ BWBCRONB | BWBCRON1 > $HOME/bwbcronb.log  
30 19 * * 1-5 BWBCRONB | BWBCRON1 > /u/userid/bwbcronb.log ;
```

Consulte *UNIX System Services Command Reference* (SA22-7802) y *UNIX System Services Planning* (GA22-7800) para obtener más información sobre los servicios de CRON disponibles en formato CRONTAB.

De forma alternativa, utilice el mandato de z/OS UNIX en línea **man**:

- `man cron`
- `man crontab`
- `man at`

Muestras de trabajos de construcción de CRON

Esta sección muestra las muestras de trabajo BWBCRON1, BWBCRONB y BWBCRONP tal como se suministran en la biblioteca SFEKSAMV.

El ejemplo siguiente muestra el código BWBCRON1.

```

/* REXX */
/* Personalizar STEPLIB, _SCLMDT_CONF_HOME y _SCLMDT_WORK_HOME */
/*
STEPLIB debería reflejar las bibliotecas de carga para
Rational Developer for System z.
Si estos conjuntos de datos residen en LINKLIST, establezca STEPLIB en '' .
*/
STEPLIB = 'FEK.SFEKLOAD:FEK.SFEKAUTH'
/*
Las variables de entorno _SCLMDT_CONF_HOME y _SCLMDT_WORK_HOME determinan los directorios HOME
en los que residen los archivos de configuración y el área de trabajo para
SCLM Developer Toolkit. Consulte el archivo de configuración de Rational Developer for System z
/etc/rdz/rsed.envvars para obtener el valor correcto.
*/
_SCLMDT_CONF_HOME = '/etc/rdz/sclmdt'
_SCLMDT_WORK_HOME = '/var/rdz'

/* USO DE EJEMPLO */
COMMAND : BWBCRONB|BWBCRON1 > BWBCRONB.log
(pasa la lista de parámetros de construcción a BWBCRON1 y las salidas a BWBCRONB.log)
*/
/* NO MODIFICAR LO QUE SIGUE */

CALL ENVIRONMENT 'STEPLIB',STEPLIB
CALL ENVIRONMENT '_SCLMDT_CONF_HOME',_SCLMDT_CONF_HOME
CALL ENVIRONMENT '_SCLMDT_WORK_HOME',_SCLMDT_WORK_HOME

CALL BWBINT

EXIT

```

Figura 18. BWBCRON1 - Exej de construcción de CRON

El ejemplo siguiente muestra el código BWBCRONB.

```

/* REXX */
/* ARCHIVO DE PARÁMETROS DE CONSTRUCCIÓN DE MUESTRA PARA CONSTRUCCIONES INICIADAS POR CRON */
/* Actualice los parámetros de construcción que aparecen a continuación */
/* si el parámetro necesario está en blanco, establézcalo como '' */
FUNCTION = 'BUILD'
PROJECT = '' /* Proyecto SCLM */
PROJDEF = '' /* Definición de proy. alt.*/
TYPE = '' /* Tipo SCLM */
MEMBER = '' /* Nombre de miembro SCLM */
GROUP = '' /* Grupo SCLM */
GROUPBLD = '' /* Constr. en grupo */
REPDGRP = '' /* Grupo de desarrollo de usuarios */
BLDREPT = 'Y' /* Generar informe de constr.*/
BLDLIST = 'Y' /* Generar lista si error */
BLDMSG = 'Y' /* Generar mensajes de constr.*/
BLDScope = 'N' /* Construir ámbito E/L/N/S */
BLDMODE = 'C' /* Modalidad de constr. C/F/R/U */
BLDMSGDS = '' /* Conjunto de datos de mensaje */
BLDRPTDS = '' /* Conjunto de datos de informe */
BLDLSTDS = '' /* Conjunto de datos de lista */
BLDEXTDS = '' /* Conjunto de datos de salida */
SUBMIT = 'BATCH' /* En línea o por lotes*/
/*
Si ejecuta en BATCH y requiere JOBCARD JCL para alterar temporalmente
el valor predeterminado, añadir hasta 4 líneas de líneas JOBCARD.
Especificar en el formato de LINE. e incluir la variable LINECNT
para el número de líneas.
*/
LINECNT = 2
LINE.1 = '//SCLMBLD JOB (XXX),SCLMBUILD,MSGCLASS=X,NOTIFY=&SYSUID,'
LINE.2 = '// CLASS=A,REGION=0M'

/* NO ALTERAR LA VARIABLE DE CONSTRUCCIÓN DE PARÁM. A CONTINUACIÓN */
PARM1 = 'SCLMFUNC='FUNCTION'&PROJECT='PROJECT'&PROJDEF='PROJDEF||',
        '&TYPE='TYPE'&MEMBER='MEMBER'&GROUP='GROUP'&GROUPBLD='GROUPBLD||',
        '&REPDGRP='REPDGRP'&BLDREPT='BLDREPT'&BLDLIST='BLDLIST||',
        '&BLDMSG='BLDMSG'&BLDScope='BLDScope'&BLDMODE='BLDMODE||',
        '&BLDMSGDS='BLDMSGDS'&BLDRPTDS='BLDRPTDS'&BLDLSTDS='BLDLSTDS||',
        '&BLDEXTDS='BLDEXTDS'&SUBMIT='SUBMIT'
/* produce como salida la serie de parámetro como entrada de BWBCRON1 */
SAY PARM1
If (SUBMIT = 'BATCH') & (LINECNT > 0) then
Do
SAY '<JOBCARD>'
Do i = 1 to LINECNT
SAY LINE.i
Final
SAY '</JOBCARD>'
Fin

```

Figura 19. BWBCRONB - Archivo de parámetros de construcción

El ejemplo siguiente muestra el código BWBCRONP.


```

/* ARCHIVO DE PARÁMETROS DE PROMOCIÓN DE MUESTRA UTILIZADO PARA PROMOCIONES INICIADAS POR CRON */
/* Actualice los parámetros de promoción que aparecen a continuación entre comillas. */
/* si el parámetro necesario está en blanco, establézcalo como '' */
FUNCTION = 'PROMOTE'
PROJECT = '' /* Proyecto SCLM */
PROJDEF = '' /* Definición de proy. alt.(opc.) */
TYPE = '' /* Tipo SCLM */
MEMBER = '' /* Nombre de miembro SCLM */
GROUP = '' /* Grupo SCLM */
GROUPPRM = '' /* Promover en grupo (opc.) */
REPDGRP = '' /* Grupo de desarrollo de usuarios */
PRMREPT = 'Y' /* Generar informe de promoción */
PRMMSG = 'Y' /* Generar mensajes de promoción */
PRMSCOPE = 'N' /* Ámbito de promoción E/L/N/S */
PRMMODE = 'C' /* Modalidad de promoción C/F/R/U */
PRMSGDS = '' /* Conjunto de datos de mensaje */
PRMRPTDS = '' /* Conjunto de datos de informe */
PRMEXTDS = '' /* Conjunto de datos de salida */
SUBMIT = 'BATCH' /* En línea o por lotes */
/*
Si ejecuta en BATCH y requiere JOBCARD JCL para alterar temporalmente
el valor predeterminado, añadir hasta 4 líneas de líneas JOBCARD.
Especificar en el formato de LINE. e incluir la variable LINECNT
para el número de líneas.
*/
LINECNT = 2
LINE.1 = '//SCLMBLD JOB (XXX),SCLMBUILD,MSGCLASS=X,NOTIFY=&SYSUID,'
LINE.2 = '// CLASS=A,REGION=0M'

/* NO ALTERAR LA VARIABLE DE PROMOCIÓN DE PARÁM. A CONTINUACIÓN */
PARM1 = 'SCLMFUNC='FUNCTION'&PROJECT='PROJECT'&PROJDEF='PROJDEF||',
        '&TYPE='TYPE'&MEMBER='MEMBER'&GROUP='GROUP'&GROUPPRM='GROUPPRM||',
        '&REPDGRP='REPDGRP'&PRMREPT='PRMREPT||',
        '&PRMMSG='PRMMSG'&PRMSCOPE='PRMSCOPE'&PRMMODE='PRMMODE||',
        '&PRMSGDS='PRMSGDS'&PRMRPTDS='PRMRPTDS'&PRMEXTDS='PRMEXTDS||',
        '&SUBMIT='SUBMIT
/* produce como salida la serie de parámetro como entrada de BWBCRON1 */
SAY PARM1
If (SUBMIT = 'BATCH') & (LINECNT > 0) then
Do
SAY '<JOBCARD>'
Do i = 1 to LINECNT
SAY LINE.i
Final
SAY '</JOBCARD>'
Fin

```

Figura 20. BWBCRONP - Archivo de parámetros de promoción

Apéndice A. Visión general de SCLM

SCLM Developer Toolkit, una función de IBM Rational Developer for System z, proporciona los medios para que las aplicaciones distribuidas escritas en Eclipse puedan gestionarse y construirse utilizando Software Configuration and Library Manager (SCLM), el sistema de gestión de código fuente z/OS de IBM.

El lenguaje y las herramientas utilizadas por los usuarios distribuidos y de sistema principal son tan variados como los entornos que utilizan. La identificación y la comprensión de conceptos claves de ambos entornos permite integrarlos en una estructura cohesiva de forma satisfactoria.

En cuanto a la estructura de aplicación, SCLM Developer Toolkit es una serie de conectores de Eclipse con el correspondiente código host de z/OS que permite el uso de transportes HTTP y RSE. Desde un punto de vista operativo, un desarrollador de Eclipse registra un proyecto de espacio de trabajo con SCLM. Los archivos del proyecto pueden añadirse a un proyecto SCLM, registrarse, examinarse y, opcionalmente, construirse y desplegarse. Todos estos servicios se activan mediante el menú Acciones de equipo. Desde el punto de vista de los administradores de SCLM, éstos pueden crear proyectos, tipos, lenguajes y conversores de construcción asociados. Características, tales como códigos de cambio y autorización, dependen de los requisitos.

Conceptos de SCLM

Desde la perspectiva de los desarrolladores Java/J2EE, los siguientes conceptos ayudan a describir lo que es SCLM:

Denominación de archivos

El sistema de archivos de z/OS sólo soporta longitudes de nombre de archivo de ocho caracteres. La interfaz de Developer Toolkit proporciona un servicio de conversión que da soporte a los convenios de nombres largos habituales en Java. Hay archivos específicos de SCLM que deben atenerse a la restricción de denominación. Están principalmente relacionados con la estructura ARCHDEF descrita en “Formato ARCHDEF J2EE” en la página 10.

Tipo

Cada archivo (denominado miembro en terminología SCLM) que se almacena en un proyecto SCLM se almacena en un conjunto de datos. El conjunto de datos en el que se almacena el archivo se identifica por el tipo SCLM. El tipo forma parte del nombre del conjunto de datos, constituido por SCLM Proyecto.Grupo.Tipo en lo que concierne a SCLM con un lenguaje asociado al mismo. Pueden existir muchos tipos definidos en un proyecto SCLM. Estos tipos permiten que dos archivos con el mismo nombre se puedan almacenar en el mismo proyecto SCLM. Cada proyecto puede contener muchos tipos. Al utilizar el tipo, se pueden almacenar varios proyectos de Eclipse en un proyecto SCLM, aunque cada proyecto IDE (entorno de desarrollo integrado) potencialmente tenga un archivo .project y .classpath. Si no distinguimos estos archivos utilizando el tipo, sólo existirá una copia de estos archivos en SCLM.

El administrador de SCLM es el responsable de la definición de tipos SCLM. Cuando comparta un proyecto con SCLM, deberá saber qué tipo va a utilizar al almacenar objetos en SCLM.

Lenguaje

Cuando añada un archivo a SCLM, deberá almacenarlo con una definición de lenguaje determinada. Una vez más, el administrador de SCLM es el responsable de la definición de lenguaje. Esta definición controla el comportamiento de SCLM mientras se transfieren los archivos hacia y desde el sistema principal. Utilizando la definición de lenguajes se puede definir si un determinado tipo de archivo se ha convertido a un nombre largo, si se ha almacenado como un objeto binario o si se ha convertido a ASCII o EBCDIC (codificación z/OS nativa). Por ejemplo, es posible que una definición de lenguaje de JAVABIN esté relacionada con un objeto binario convertido a un nombre largo. De forma alternativa, se puede definir WEBHTML de forma que represente a un archivo de texto convertido a nombre largo que va a almacenarse en ASCII. El número de definiciones de lenguaje se define por proyecto. Es necesario comprender qué lenguaje se debe utilizar para garantizar que el archivo se almacene y se pueda recuperar correcta desde SCLM. El lenguaje también define la forma en que SCLM construye un objeto.

Propiedades de SCLM

Cualquier archivo que esté bajo el control de SCLM tendrá una serie de propiedades asociadas al mismo. Estas propiedades son efectivamente el mecanismo de correlación entre el archivo IDE y sus respectivas propiedades SCLM. Cuando se realizan llamadas de servicio a SCLM, se leen estos datos para formular los parámetros de servicio adecuados. Puede visualizarlos desde el menú Propiedades resaltando un miembro controlado por SCLM desde Eclipse.

Estructura de proyectos SCLM

Cuando comparta un proyecto con SCLM, también deberá designar a qué grupo de desarrollo pertenece. Las estructuras de proyecto SCLM son jerárquicas por naturaleza.

Inicialmente, el código se almacena a nivel de DEVELOPMENT (desarrollo). Una vez construido y probado de forma satisfactoria, se puede promover a TEST (prueba). Si la siguiente prueba también resulta satisfactoria, se puede promover a PRODUCTION (producción). Normalmente, esto representa al producto desarrollado. Si se detecta un defecto en el código de nivel de producción, los archivos que deben editarse para arreglar el defecto se copiarán en el nivel de desarrollo y el proceso de construcción volverá a empezar. Todos los códigos que forman la aplicación no se copian a nivel de desarrollo. SCLM conserva una pista de los elementos que forman la construcción y del nivel en el que se han almacenado.

Dentro del nivel de desarrollo pueden haber varios grupos. Esto permite separar el código almacenado a nivel de desarrollo. SCLM también proporciona controles para determinar el comportamiento del código almacenado en distintos grupos de desarrollo en cuanto a la capacidad de promoción.

ARCHDEF

La estructura del proyecto IDE normalmente se compone de uno o varios proyectos IDE. Al almacenar cada uno de estos proyectos IDE en un tipo SCLM distinto, esta estructura se conserva. El archivo ARCHDEF define efectivamente los archivos que forman un proyecto IDE. Cada proyecto SCLM puede tener varias

ARCHDEF. Es posible que una ARCHDEF haga referencia a otras ARCHDEF de forma que esta estructura de proyectos de varios IDE se pueda definir para el proceso de construcción, siendo la ARCHDEF el medio principal para definir una lista de construcciones para SCLM. La analogía más parecida es la de un proceso "make". La ARCHDEF lista los archivos que forman la construcción y, además, especifica un script de construcción que permitirá especificar la ubicación de JAR o clases externos. Para obtener más información, consulte la sección del manual del usuario del sistema de ayuda en línea.

Conceptos de JAVA/J2EE

Cuando se crea un proyecto IDE en el espacio de trabajo, se genera automáticamente un archivo de descripción de proyecto que se almacena bajo el nombre **.project**. Este documento XML contiene descripciones de todos los "creadores" o "naturalezas" asociados al proyecto. Los primeros son creadores de proyectos incrementales que crean estados de construcción basados en el contenido del proyecto. Puesto que el contenido del proyecto cambia, este archivo se irá actualizando. Las naturalezas definen y gestionan la asociación entre un proyecto determinado y un conector o característica en particular.

El archivo **.classpath** describe la vía de acceso que se utiliza para buscar JAR y clases externos referenciados por el código fuente del proyecto IDE. La función equivalente durante una construcción mediante SCLM Developer se define con la directiva **CLASSPATH_JARS** en los scripts de construcción Ant. Esta directiva describirá la vía de acceso en el host de z/OS que se utiliza para buscar JAR y clases externos referenciados por el código fuente del proyecto IDE.

Tanto **.classpath** como **.project** se utilizan para conservar la configuración del proyecto IDE de forma que se pueda volver a crear en otro espacio de trabajo. Por este motivo, se recomienda que ambos se comprueben en SCLM como parte del proyecto IDE.

Un aspecto importante del desarrollo del proyecto, concretamente de los proyectos J2EE, es que se pueden crear varias formas distintas de ejecutables de aplicación. Los ejecutables de proyecto Java a menudo se empaquetan como archivos JAR, WAR, RAR o EAR.

Los archivos JAR normalmente se refieren a aplicaciones Java estándar o a Enterprise Java Beans (EJB).

Los archivos WAR se crean para aplicaciones web. Normalmente están compuestos de servlets Java, JSP y archivos HTML. Las aplicaciones WAR a menudo son procesadores frontales (interfaces) de aplicaciones basadas en la web. Estas aplicaciones pueden hacer referencia a otros JAR tales como EJB para servicios específicos. Cada archivo WAR contiene un archivo **web.xml**. Éste describe la composición de la aplicación WAR en términos de Java, HTML y los servicios de biblioteca que utiliza.

El desarrollo de archivos RAR actualmente no se soporta en SCLM Developer Toolkit.

Los archivos EAR representan aplicaciones empresariales. Estas aplicaciones están compuestas por archivos JAR y WAR. En lenguaje J2EE, la creación del archivo EAR es el ensamblaje de sus archivos JAR y WAR constituyentes. Este método de ensamblaje permite crear aplicaciones EAR que, de hecho, están formadas por componentes específicos (JAR/WAR). La composición real de la aplicación se

describe en el archivo `application.xml`. El propio archivo EAR no es un objeto ejecutable autónomo. El archivo EAR debe instalarse en un contenedor J2EE como Websphere Application Server (WAS). La instalación del archivo EAR se denomina despliegue. Se puede acceder a una aplicación EAR desplegada mediante el entorno WAS. El despliegue es un proceso independiente del de la construcción. El despliegue implica la instalación física de la aplicación EAR.

Por consiguiente, el desarrollo de aplicaciones J2EE es posible que también implique el desarrollo de varios componentes independientes tales como archivos WAR y JAR. A continuación, estos archivos se ensamblan en un archivo EAR. El archivo EAR ya estará listo para el despliegue en un contenedor J2EE (por ejemplo, WAS) para la operación.

Dentro del espacio de trabajo de Eclipse, los proyectos son efectivamente adyacentes; es decir, que dentro del entorno de IDE pueden referirse efectivamente a otros proyectos IDE en términos de empaquetado con gran facilidad. Dentro de SCLM, cada uno de estos proyectos IDE (por ejemplo, proyectos WAR, JAR y EAR) deben correlacionarse en un único proyecto SCLM, cada uno de ellos diferenciado mediante el uso de un tipo SCLM distinto. El motivo es que existen nombres de archivos comunes utilizados en muchos proyectos IDE tales como `.project`, `.classpath`, `web.xml` y `application.xml`, por lo que el uso de tipos independientes permite que existan en el mismo proyecto SCLM estas partes con los mismos nombres. Para obtener más información sobre correlaciones, consulte “Correlación de proyectos J2EE con SCLM” en la página 18.

Desde el punto de vista de SCLM, el desarrollo de una aplicación EAR se referencia mejor si se utiliza una estructura ARCHDEF de nivel superior. Dentro de SCLM, las ARCHDEF de nivel superior, en muchos proyectos SCLM denominadas *paquete*, constituyen el vértice de la estructura de ARCHDEF, seguidas por la aplicación EAR y referencias de nivel inferior (archivos WAR y JAR) que forman la aplicación EAR. Esta estructura permite utilizar construcciones tanto a nivel superior como inferior, además de construcciones completas o condicionales. De este modo, las ARCHDEF proporcionan una forma de definir elementos de proyecto J2EE dentro del proyecto SCLM.

Apéndice B. Tabla de conversión de nombres largos/cortos

Actualmente, SCLM base no soporta el uso del almacenamiento de código con nombres de archivo (miembro) que superen los ocho caracteres.

El código, como Java y otro código de cliente PC, tiene intrínsecamente longitudes de nombre mucho más largas e incluso incorpora información de vía de acceso (empaquetado) como parte del nombre. Esto hace que sea necesario que, cuando el código tenga partes del nombre que superen los ocho caracteres, se deba recurrir al programa de utilidad de conversión de nombres largos/cortos para permitir que estas partes se almacenen dentro de SCLM con una longitud de nombre de ocho caracteres (o menos).

Una tabla de conversión de nombres largos a nombres cortos almacena los nombres largos (nombre real) con sus correspondientes nombres cortos (tal como se han almacenado en SCLM). Es posible controlar y acceder a estas tablas mediante SCLM y se guardan en un conjunto de datos VSAM. Esta funcionalidad se ha introducido en SCLM con el arreglo temporal del programa (PTF) que aborda el APAR OA11426 para z/OS 1.4 y posteriores. Para z/OS 1.8 y posteriores, este arreglo temporal del programa no es necesario.

El algoritmo de conversión efectúa los pasos siguientes:

1. El prefijo de conversión consta de los dos primeros caracteres (mayúsculas) del nombre largo del programa/archivo (es decir, el último nombre de archivo después de del carácter "/" en el formato de varios paquetes). Si los dos primeros caracteres no son válidos como prefijo para un nombre de miembro de host (porque contienen caracteres especiales no válidos), el prefijo es "XX". En casos especiales, tales como un nombre alfabético de un único carácter (/u/test/A o /u/test/A.java), también se asigna el prefijo "XX".
2. A los seis últimos caracteres se les asignan numéricamente el siguiente número secuencial disponible en la tabla de conversión.

Ejemplo

Nombre largo	Nombre corto en el PDS o PDSE de SCLM
com/ibm/workbench/testprogram.java	TE000001
source/plugins/Phantom/.classpath	XX000001

Resumen técnico del programa de conversión de SCLM

El programa FLMLSTRN de SCLM se creó para leer y actualizar la tabla de conversiones VSAM. SCLM Developer Toolkit utiliza este programa para leer y actualizar nombres largos y cortos correlativos.

El archivo VSAM utilizado para almacenar la tabla de conversiones es un KSDS de longitud variable con un índice y una vía de acceso alternativos definidos. Se suministra un trabajo de muestra en SCLM para asignar este archivo VSAM.

La estructura interna del clúster de VSAM es:

```

1 ls_record,
3 ls_short_name char(08),
3 ls_lngname_key char(50),
3 ls_long_name char(1024);

```

El lenguaje de control de trabajos (JCL) de muestra para asignar el archivo VSAM de conversión de nombres largos/cortos se puede ver en el paso 6: Configuración del archivo VSAM de tabla de nombres largos/cortos.

Nota: La siguiente información técnica sobre las llamadas a la función de tabla de conversiones SCLM se suministra únicamente a título informativo y no es necesaria para habilitar ninguna funcionalidad de SCLM Developer Toolkit.

El programa FLMLSTRN se invoca con el servicio ISPF SELECT con uno de los parámetros listados en la Tabla 12.

Sintaxis:

```
"SELECT PGM(FLMLSTRN) PARM(keyword)"
```

Ejemplo de invocación:

```
"SELECT PGM(FLMLSTRN) PARM(TRANSLATE)"
```

Tabla 12. Parámetros de conversión de nombres largos/cortos.

Registro de palabras clave	Proceso	Descripción
FINDLONG	Único	Búsqueda de un nombre largo para un nombre corto determinado
FINDSHORT	Único	Búsqueda de un nombre corto para un nombre largo determinado
TRANSLATE	Único	Búsqueda de un nombre corto, si existe, o asignación de un nombre corto nuevo si no existe
MIGRATE	Varios	Búsqueda de varios nombres largos
IMPORT	Varios	Búsqueda de varios nombres cortos

Proceso de registros de nombres largos/cortos único

Proceso FINDLONG

- El clúster VSAM asignado a DD LSTRANS se abre en modalidad de lectura.
- El nombre corto se recupera desde la variable FLMLSSHR de ISPF y se utiliza para leer el archivo VSAM.
- Si no se encuentra el registro, se devuelve un mensaje mediante la variable FLMLSERR de ISPF declarando que no se ha encontrado el nombre largo.
- Si se ha encontrado el nombre largo, éste se devuelve en la variable FLMLSLNG de ISPF.
- El clúster VSAM se cierra.

Proceso FINDSHORT

- La vía de acceso VSAM asignada a DD LSTRNPTH se abre en modalidad de lectura.
- El nombre largo se recupera desde la variable FLMLSLNG de ISPF.
- Los últimos 50 bytes del nombre largo se utilizan para leer la vía de acceso.

- Si no se devuelve un registro, se devuelve un mensaje mediante la variable FLMLSERR de ISPF declarando que no se ha encontrado el nombre corto.
- Si se devuelve un registro, el nombre largo del registro VSAM se comprueba con el nombre largo de la variable FLMLSLNG de ISPF.
- Si no hay correspondencia, se leen los registros VSAM y se comparan hasta que ls_lngname_key no coincida o se encuentre el nombre largo.

Nota: La ls_lngname_key permite duplicados porque se puede tener un registro VSAM con la misma ls_lngname_key pero con un nombre largo distinto.

- Si se ha encontrado el nombre corto, éste se devuelve en la variable FLMLSSHR de ISPF.
- La vía de acceso VSAM se cierra.

Proceso TRANSLATE

El proceso es el mismo que para FINDSHORT, tal como se detalla a continuación:

- Si se encuentra el nombre corto, no se realizará ningún proceso ulterior.
- Si no se encuentra el nombre corto, el clúster asignado a DD LSTRANS se abre en modalidad de actualización.
- El nombre de archivo se determina buscando la última '/' o '\' del nombre largo.
- Los dos primeros bytes del nombre de archivo se utilizan para buscar el registro de prefijos de archivos VSAM que contiene un número.
- El prefijo y el número del archivo se utilizarán para generar el nombre corto (por ejemplo, PR000123).
- El nombre corto generado (PR000123) se utiliza para comprobar el archivo VSAM y ver si utilizando el nombre corto.
- Si es así, el número de prefijo se incrementa y se vuelve a comprobar el nombre corto.
- El proceso continua hasta que se encuentra un nombre corto que no se esté utilizando.
- El registro de prefijos se actualiza y, a continuación, se añade el nuevo registro de conversión.
- El nombre corto se devuelve en la variable FLMLSSHR de ISPF.
- El clúster VSAM se cierra.

Proceso de registros de nombres largos/cortos múltiple

MIGRATE e IMPORT son funciones que se han introducido para mejorar el rendimiento cuando se convierten grandes cantidades de nombres largos (MIGRATE) o se buscan grandes cantidades de nombres cortos (IMPORT).

Ambas funciones, "MIGRATE" e "IMPORT", leen un archivo secuencial bloqueado variable con LRECL=1036, que está asignado como DD LSTRNPRC.

Antes de la invocación, este archivo contendrá los nombres cortos o los nombres largos, dependiendo de la función llamada y en el formato y la columna correctos.

Después de la invocación, LSTRNPRC contendrá tanto el nombre corto como el nombre largo correlativo.

El formato del archivo es el siguiente:

```

1 pr_record,
3 pr_short_name  char(08),
3 pr_long_name   char(1024);

```

Proceso IMPORT

- El clúster VSAM asignado a DD LSTRANS se abre en modo de lectura y el archivo de proceso asignado a DD LSTRNPRC se abre para su actualización.
- En cada uno de los registros del archivo de proceso se utiliza el nombre corto para leer el archivo de conversiones VSAM. Si se encuentra un registro, el archivo de proceso se actualiza con el nombre largo.
- Se cierran el clúster VSAM/los archivos de proceso.

Proceso MIGRATE

- El clúster VSAM asignado a DD LSTRANS se abre en modo de lectura y el archivo de proceso asignado a DD LSTRNPRC se abre para su actualización.
- Para cada uno de los registros en el archivo de proceso se utiliza el nombre largo para leer el archivo VSAM. Si se encuentra un registro, el registro de archivos de proceso se actualiza con su nombre corto correspondiente; de lo contrario, se abre LSTRANS en modalidad de actualización para añadir entradas de nombres largos/cortos y el nuevo nombre corto generado se vuelve a escribir en el archivo LSTRNPRC.
- Se cierran el clúster VSAM/los archivos de proceso.

El siguiente ejemplo muestra el código REXX de muestra para invocar al proceso de conversión de nombres largos/cortos.

```

/* REXX *****/
/* Muestra para convertir un nombre largo en un nombre corto */
/*****/
Address TSO
"FREE FI(LSTRANS)"
"FREE FI(LSTRNPTH)"
"ALLOC DD(LSTRANS) DA('FEK.#CUST.LSTRANS.FILE') SHR REUSE"
"ALLOC DD(LSTRNPTH) DA('FEK.#CUST.LSTRANS.FILE.PATH') SHR REUSE"
/* Crear nombre corto para nombre largo com/ibm/phantom.txt */
FLMLSLNG = "com/ibm/phantom.txt"
Address ISPEXEC "VPUT (FLMLSLNG) PROFILE"
Address ISPEXEC "SELECT PGM(FLMLSTRN) PARM(TRANSLATE)"
LSRC=RC
If LSRC > 0 Then
Do
  Address ISPEXEC "VGET (FLMLSERR,FLMLSER1) PROFILE"
  Say "LS ERROR LINE 1 ==>" FLMLSERR
  Say "LS ERROR LINE 2 ==>" FLMLSER1
  Return
Final
Else
Do
  Address ISPEXEC "VGET (FLMLSSHR,FLMLSLNG) PROFILE"
  Say " Shortname = " FLMLSSHR
  Say " Longname = " FLMLSLNG
Final
Address TSO
"FREE FI(LSTRANS)"
"FREE FI(LSTRNPTH)"

```

Figura 21. REXX de muestra para la invocación del módulo de conversión

Apéndice C. API de SCLM Developer Toolkit

Este apéndice documenta la Interfaz de programación de aplicaciones (API) para los servicios de host de SCLM Developer Toolkit. La API utiliza un formato de solicitud y respuesta basado en XML y debe accederse a la misma mediante z/OS UNIX Systems Services.

La API permite que los usuarios utilicen los servicios de host de SCLM Developer Toolkit con su propio cliente/conector y capa de transporte, si así lo desean.

Se suministra un programa Java de muestra (con entrada y salida) que accede a la API de los servicios de host SCLMDT utilizando un servidor HTTP como mecanismo de transporte.

Muchas de las solicitudes de función se basan en los servicios de host SCLM actuales y podrá obtener más información acerca de valores de parámetros parecidos para funciones comunes en la Guía de SCLM, así como referencias sobre el release de z/OS relevante.

Nota: Esta API documenta los servicios de host que normalmente utiliza el cliente de SCLM Developer Toolkit Client. Por ello, muchas de las funciones pueden resultar menos adecuadas para que los utilice el cliente o requerirán una validación adicional por parte de éste.

Formato de invocación

El siguiente mandato de muestra ilustra cómo se pueden invocar los servicios de host SCLMDT:

```
cat sclmdt_request.xml | BWBXML > sclmdt_response.xml
```

cat	Mandato de z/OS UNIX para visualizar archivos de texto
sclmdt_request.xml	Archivo de entrada XML con la solicitud de usuario
	Mandato de z/OS UNIX para conducir la salida del mandato anterior como entrada del siguiente
BWBXML	Script de conversión XML, residente en el directorio de Developer for System z /bin, que invoca a la interfaz de servicios SCLMDT
>	Mandato de z/OS UNIX para redireccionar la salida del mandato anterior hacia un archivo
sclmdt_response.xml	Archivo de salida XML que contiene la respuesta de servicio

Notas:

1. La variable de entorno PATH debe contener la ubicación del directorio de BWBXML, por ejemplo:

```
export PATH=/usr/lpp/rdz/bin:$PATH
```
2. Cuando se utiliza HTTP como mecanismo de transporte:
 - El script de interfaz BWBXML se puede invocar como un script CGI (tipo EXEC en las directivas del servidor HTTP).
 - La solicitud se lee como STDIN a través de BWBXML, de forma que se puede pasar al script CGI como una solicitud POST (consulte el programa de muestra).

Esquema XML para mandatos SCLMDT

El ejemplo siguiente muestra el esquema XML para mandatos SCLMDT referenciados en el archivo de entrada XML. Este ejemplo también está disponible como miembro BWBXSD1 en la biblioteca de muestra FEK.SFEKSAMV.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="SCLMDT-INPUT">
  <xs:complexType>
    <xs:all>

      <xs:element name="SERVICE-REQUEST">
        <xs:complexType>
          <xs:all>

            <xs:element name="service">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <!-- Especifica la llamada de servicio TSO o ISPF nativa -->
                  <xs:enumeration value="ISPF"/>
                  <xs:enumeration value="TSO"/>
                  <xs:enumeration value="SCLM"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>

            <xs:element name="session" minOccurs="0">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <!-- Valor predeterminado NONE : La sesión termina tras la llamada de servicio -->
                  <xs:enumeration value="NONE"/>
                  <!-- La sesión ISPF reutilizable permanece activa entre llamadas -->
                  <xs:enumeration value="REUSE"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>

            <!-- Utilizar perfil ISPF existente en llamada -->
            <xs:element name="ispprof" type="xs:string" minOccurs="0"/>

            <!-- Mandato TSO/ISPF de formato libre -->
            <xs:element name="command" type="xs:string" minOccurs="0"/>

            <!-- Lista de todas las funciones DT SCLM disponibles -->

            <!-- sclmfunc : Función SCLM seleccionada -->
            <xs:element name="sclmfunc" minOccurs="0">
              <xs:simpleType>
                <xs:restriction base="xs:string">

                  <xs:enumeration value="BUILD"/> <!-- Función de construcción SCLM -->
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <!-- Parámetros de construcción : project,projdef,group,repdgrp,type,member,bldmode,bldlist,
            bldrept,bldmsg,bldmsgds,bldlist,bldlstds,bldextds,groupbld,submit -->

            <xs:enumeration value="PROMOTE"/> <!-- Función de promoción SCLM -->
            <!-- Parámetros de promoción : project,projdef,group,repdgrp,type,member,prmmode,prmrept,
            prmmsg,prmmsgds,prmextds,groupprm,submit -->

            <xs:enumeration value="EDIT"/> <!-- Miembro EDIT -->
            <!-- Parámetros de edición : project,projdef,group,repdgrp,type,member,lang -->
            <!-- Nota: Parámetro transferido a DTinstall/WORKAREA -->

            <xs:enumeration value="BROWSE"/> <!-- Miembro de navegación -->
            <!-- Parámetros de navegación : project,projdef,group,repdgrp,type,member,lang -->
            <!-- Nota: Parámetro transferido a DTinstall/WORKAREA -->

            <xs:enumeration value="SAVE"/> <!-- Guardar miembro editado -->
            <!-- Parámetros de guardado : project,projdef,group,repdgrp,type,member,lang -->
            <!-- Nota: Miembro recibido desde DTinstall/WORKAREA -->

            <xs:enumeration value="DELETE"/> <!-- Miembro de supresión SCLM -->
            <!-- Parámetros de supresión : project,projdef,group,repdgrp,type,member,delflag -->

            <xs:enumeration value="UNLOCK"/> <!-- Miembro de desbloqueo SCLM -->
            <!-- Parámetros de desbloqueo : project,projdef,group,repdgrp,type,member -->

            <xs:enumeration value="DEPLOY"/> <!-- Función de despliegue J2EE -->

```

Funciones y parámetros de solicitud

Formato de función

El ejemplo siguiente muestra la estructura básica del archivo de entrada XML, donde #function representa la función llamada y #parameter y #value son un parámetro y su valor.

```
<?xml version="1.0"?>
<SCLMDT-INPUT
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="sclmdt.xsd">
  <SERVICE-REQUEST>
    <service>SCLM</service>
    <session>NONE</session>
    <sclmfunc>#function</sclmfunc>
    <#parameter>#value</#parameter>
    ...
  </SERVICE-REQUEST>
</SCLMDT-INPUT>
```

Figura 23. Estructura básica para el archivo de entrada XML

Nota: Si un parámetro se especifica más de una vez, se utilizará la definición de la última instancia.

La lista siguiente contiene algunas observaciones generales sobre los parámetros y el modo en que se han documentado en esta publicación:

- Los parámetros no son posicionales.
- Los corchetes ([]) indican un parámetro opcional para la función.
- Las llaves ({}) indican una lista de valores posibles para el parámetro.
- Una barra horizontal (|) separa varios valores de una lista. El valor subrayado, si lo hay, es el valor predeterminado.
- Las palabras clave en mayúsculas representan constantes que se deben codificar tal como están; las palabras clave en minúsculas representan marcadores de posición que se deben sustituir por valores personalizados.
- Cada parámetro necesita un valor, pero sólo se documentan los valores que forman parte de una lista.
- Las referencias a WORKAREA se refieren al directorio WORKAREA de z/OS UNIX, ubicado en /var/rdz/sclmdt/ de forma predeterminada.
- Las referencias a member necesitan que se utilice el nombre del miembro (SCLM) con su nombre corto, a no ser que se indique explícitamente lo contrario.

Lista de funciones

- “AUTHUPD – Cambiar código de autoridad SCLM” en la página 69
- “BROWSE – Examinar miembro SCLM” en la página 70
- “BUILD – Miembro SCLM de construcción” en la página 70
- “DELETE – Suprimir un miembro SCLM” en la página 71
- “DEPLOY – Desplegar un archivo EAR J2EE” en la página 72
- “EDIT – Editar miembro SCLM” en la página 73
- “INFO – Información sobre el estado del miembro SCLM” en la página 73
- “J2EEIMP – Importar proyecto desde SCLM” en la página 74

- ## AUTHUPD – Cambiar código de autoridad SCLM

>>AUTHUPD-----GROUP-MEMBER-PROJECT-TYPE--<<

AUTHCODE
PROJDEF

BROWSE – Examinar miembro SCLM

Esta función copia un miembro SCLM en el directorio WORKAREA/userid/EDIT/ de z/OS UNIX. No se efectúa ningún bloqueo de edición en SCLM.

>>BROWSE

PROJDEF

 GROUP-MEMBER-PROJECT-TYPE-><

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

BUILD – Miembro SCLM de construcción

Esta función indica a SCLM que debe compilar, enlazar e integrar componentes de software de acuerdo con las definiciones de arquitectura de proyecto.

>>BUILD

BLDEXTDS
BLDLIST
BLDLSTD
BLDMODE
BLDMSGDS
BLDREPT
BLDRPTDS
BLDScope
PROJDEF
SUBMIT

 GROUP-GROUPBLD-MEMBER-PROJECT-REPDGRP-TYPE-><

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

GROUPBLD

Grupo SCLM - El grupo SCLM seleccionado en el que se debe construir el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[BLDEXTDS [dsn | NONE]]

Conjunto de datos de salida de construcción - NONE o el nombre del conjunto de datos que sostendrá la salida de construcción, si se utiliza una salida de construcción. Si el conjunto de datos no existe, se asignará un conjunto de datos nuevo. El valor predeterminado es NONE.

[BLDLIST [Y | N]]

Listado de construcción - Indica que los listados del conversor de construcciones sólo deben copiarse en el conjunto de datos de la lista si se produce en error. El valor predeterminado es Y.

[BLDLSTDS [dsn | NONE]]

Conjunto de datos de lista de construcción - NONE o el nombre del conjunto de datos que sostendrá los listados de construcción. Si el conjunto de datos no existe, se asignará un conjunto de datos nuevo. Los listados de construcción también se devolverán en el archivo de respuestas XML, independientemente de este valor. El valor predeterminado es NONE.

[BLDMODE [C | F | R | U]]

Modalidad de construcción - Indica la modalidad de construcción (C=condicional, F=forzada, R=informe, U=incondicional). El valor predeterminado es C.

[BLDMSGDS [dsn | NONE]]

Conjunto de datos de mensaje de construcción - NONE o el nombre del conjunto de datos que sostendrá los mensajes de construcción. Si el conjunto de datos no existe, se asignará un conjunto de datos nuevo. Los mensajes de construcción también se devolverán en el archivo de respuestas XML, independientemente de este valor. El valor predeterminado es NONE.

[BLDREPT [Y | N]]

Informe de construcción - Indica si debe generarse un informe de construcción. El valor predeterminado es Y.

[BLDRPTDS [dsn | NONE]]

Conjunto de datos de informe de construcción - NONE o el nombre del conjunto de datos que sostendrá el informe de construcción. Si el conjunto de datos no existe, se asignará un conjunto de datos nuevo. El informe de construcción también se devolverá en el archivo de respuestas XML, independientemente de este valor. El valor predeterminado es NONE.

[BLDScope [E | L | N | S]]

Ámbito de construcción - Indica el ámbito de construcción (E=ampliado, L=limitado, N=normal, S=subunidad). El valor predeterminado es N.

[PROJDEF]

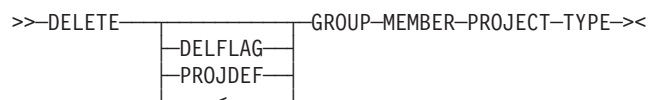
Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[SUBMIT [BATCH | ONLINE]]

Método de sometimiento - La construcción se somete en línea o por lotes. El valor predeterminado es ONLINE.

DELETE – Suprimir un miembro SCLM

Esta función suprime un miembro SCLM.



Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[DELFLAG [TEXT | ACCT | TXBM | BMAP | ALL]]

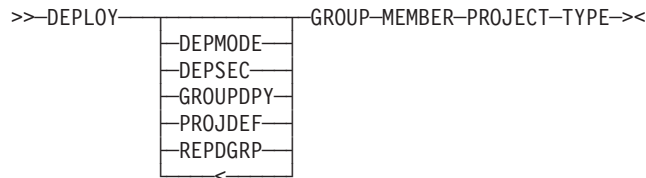
Distintivo de supresión - Indica que un texto, una cuenta, una correlación de construcción, una combinación de correlación de construcción y texto o todos ellos deben suprimirse para un miembro determinado. El valor predeterminado es ALL.

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

DEPLOY – Desplegar un archivo EAR J2EE

La función DEPLOY ejecutará el script de despliegue referenciado por un miembro para desplegar un archivo Enterprise Archive (EAR) de USS o SCLM en un Websphere Application Server (WAS). Consulte la Guía del usuario de SCLM Developer Toolkit para obtener más información acerca de cómo crear un miembro de script de despliegue.



Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[DEPSEC {Y | N}]

Despliegue seguro - Distintivo para indicar si se ha realizado una comprobación de regla de seguridad y una posible conmutación de ID de usuario sustituto para este despliegue.

[DEPMODE {R}]

Modalidad de despliegue - Si se establece en "R", el informe será de sólo lectura. No se producirá ningún despliegue.

[GROUPDPY]

Grupo de despliegue - Es el grupo SCLM desde el que se desplegará el archivo EAR (o se buscará la jerarquía del grupo si no se ha encontrado). Si se establece en 'USS', el archivo EAR se desplegará directamente desde el directorio de z/OS UNIX especificado como parte del nombre EAR en la variable EAR_FILE_NAME. Esta variable se define en el miembro de script de despliegue referenciado como 'MEMBER'.

[PROJDEF]

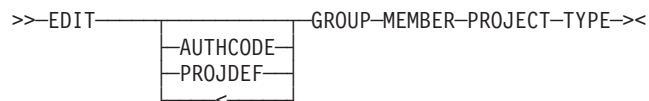
Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[REPDGRP]

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

EDIT – Editar miembro SCLM

Esta función copia un miembro SCLM en el directorio WORKAREA/userid/EDIT/ de z/OS UNIX. También crea un bloqueo de SCLM en el miembro con el ID de usuario como clave de acceso.



Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[AUTHCODE]

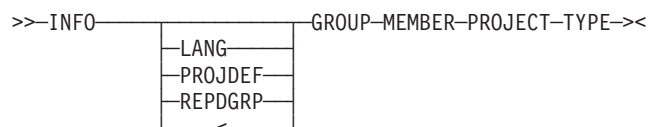
Código de autoridad de SCLM - El nuevo código de autoridad que debe asignarse al miembro.

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

INFO – Información sobre el estado del miembro SCLM

Esta función proporciona información sobre el estado del miembro SCLM.



Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[LANG]

Lenguaje SCLM - El lenguaje SCLM del miembro seleccionado.

[PROJDEF]

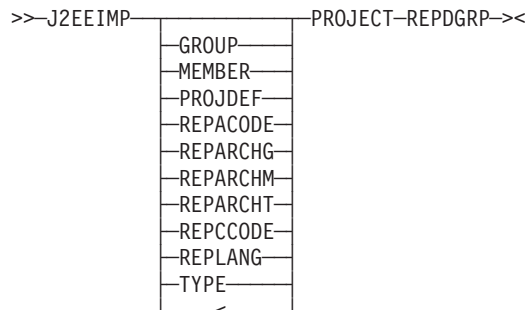
Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[REPDGRP]

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

J2EEIMP – Importar proyecto desde SCLM

Esta función importa un proyecto desde SCLM en el directorio z/OS UNIX /var/rdz/WORKAREA/userid en formato JAR (comprimido). El archivo de proyecto JAR se puede copiar posteriormente en el cliente. El nombre del archivo del proyecto se devuelve en la palabra clave J2EEFILE de la salida de función (XML).



Parámetros necesarios:

[PROJECT]

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

[REPDGRP]

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

Parámetros opcionales:

[GROUP {group | group* | *}]

Jerarquía de grupos - La jerarquía de grupos para la selección de miembros. Si se ha establecido REPARCHM, las relacionadas con dicha Archdef se ubicarán en group*.

[MEMBER {member | member* | *}]

Miembro SCLM - Miembro(s) SCLM seleccionados.

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[REPACODE]

Código de autoridad - Código de autoridad en el que se debe realizar el filtrado.

[REPARCHG]

Grupo Archdef - El grupo SCLM en el que reside el miembro Archdef.

[REPARCHM]

Miembro Archdef - Nombre del miembro Archdef que se debe seleccionar para la importación.

[REPARCHT]

Tipo Archdef - El tipo SCLM en el que reside el miembro Archdef.

[REPCODE]

Código de cambio - Código de cambio en el que se debe realizar el filtrado.

[REPLANG]

Lenguaje SCLM - Lenguaje en el que se debe realizar el filtrado.

[TYPE {type | type* | *}]

Tipo SCLM - Tipo(s) SCLM para la selección de miembros.

J2EEMIG – Migrar proyecto en SCLM

Esta función tomará un archivo comprimido (formato JAR) del directorio USS, lo extraerá y migrará el resto de miembros que existan dentro de este archivo en SCLM y convertirá los nombres largos en nombres breves cuando sea necesario.

```
>>-J2EEMIG-          GROUP-J2EEFILE-LANG-MEMBER-PROJECT-TYPE-><
|
|  ARCHAC
|  ARCHCC
|  ARCHTYPE
|  AUTHCODE
|  CCODE
|  J2EESINC
|  J2EETYPE
|  MIGMODE
|  PROJARCH
|  PROJDEF
|  SCLMREFS
|  SUBMIT
|
|  <
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

J2EEFILE

Nombre del archivo de entrada - El nombre de archivo del proyecto JAR (comprimido) que se debe importar en SCLM. El archivo JAR debe residir en el directorio z/OS UNIX /var/rdz/WORKAREA/userid.

LANG

Lenguaje SCLM - El lenguaje SCLM del miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[ARCHAC]

Código de autoridad Archdef - Establece el código de autoridad Archdef.

[ARCHCC]

Código de cambio Archdef - Establece el código de cambio Archdef.

[ARCHTYPE [ARCHDEF | archtype]]

Tipo SCLM Archdef - Establece el tipo Archdef.

[AUTHCODE]

Código de autoridad de miembro - Establece el código de autoridad de los miembros Archdef.

[CCODE]

Código de cambio de miembro - Establece el código de cambio de los miembros Archdef.

[J2EESINC]

Script de construcción SINC J2EE.- Nombre del script de construcción que está ubicado en TYPE J2EEBLD y está referenciado por la palabra clave SINC en el miembro ARCHDEF.

[J2EETYPE {JAR | WAR | EAR}]

Tipo J2EE - Especifica el JAR, WAR o EAR para el proyecto Archdef de J2EE.

[MIGMODE {FORCE}]

Modalidad de migración – FORCE sustituirá a los miembros existentes. El valor predeterminado es una migración condicional.

[PROJARCH]

Proyecto Archdef - Nombre de la Archdef que se debe actualizar o crear

[SCLMREFS]

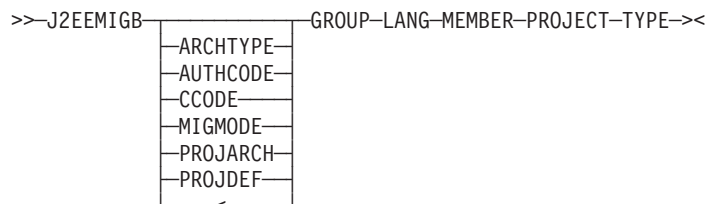
Referencias de SCLM - Archdefs o partes adicionales.

[SUBMIT [BATCH | ONLINE]]

Método de sometimiento - La migración se somete en línea o por lotes. El valor predeterminado es ONLINE.

J2EEMIGB – Migrar el proyecto por lotes en SCLM

Esta función configurará y ejecutará el trabajo por lotes (BATCH) para la migración.



Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

LANG

Lenguaje SCLM - El lenguaje SCLM del miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[ARCHTYPE]

Tipo SCLM Archdef - Establece el tipo Archdef.

[AUTHCODE]

Código de autoridad de miembro - Establece el código de autoridad de los miembros Archdef.

[CCODE]

Código de cambio de miembro - Establece el código de cambio de los miembros Archdef.

[MIGMODE {FORCE}]

Modalidad de migración – FORCE sustituirá a los miembros existentes. El valor predeterminado es una migración condicional.

[PROJARCH]

Proyecto Archdef - Nombre de la Archdef que se debe actualizar o crear.

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

JARCOPY – Copiar archivo JAR

Esta función copia un archivo JAR almacenado en SCLM en un directorio de z/OS UNIX, que se puede utilizar como un directorio de CLASSPATH.

```
>>-JARCOPY-PROJDEF-CLASSDIR-GROUP-JARNAME-PROJECT-REPDGRP-TYPE-><
```

Parámetros necesarios:

CLASSDIR

Directorio de vía de acceso - Nombre del directorio de z/OS UNIX de destino.

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

JARNAME

Nombre de archivo JAR - Nombre largo del archivo JAR de destino. El nombre corto de SCLM se busca automáticamente.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

JOBSTAT – Recuperar estado de trabajo por lotes

Esta función recupera el estado de un trabajo por lotes específico.

>>—JOBSTAT—JOBFUNC—JOBID—JOBNAME—PROJECT—<<

Parámetros:

JOBFUNC {JOBSTAT | JOBRETR}

Selección de función - Recupera el estado del trabajo (JOBSTAT) o la salida del trabajo (JOBRETR).

JOBID

ID de trabajo - Número de trabajo del trabajo por lotes.

JOBNAME

Nombre de trabajo - Nombre del trabajo por lotes.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

LRECL – Recuperar LRECL de conjunto de datos SCLM

Esta función recupera la longitud de registro lógico de un conjunto de datos SCLM.

>>—LRECL——GROUP—PROJECT—REPDGRP—TYPE—<<

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

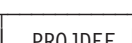
Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

MIGDSN – Listar conjuntos de datos y miembros que no son de SCLM

Esta función lista conjuntos de datos y miembros seleccionados que no se gestionan mediante SCLM (para la posterior migración en SCLM).

>>—MIGDSN——MIGDSN—MIGMEM—PROJECT—<<

Parámetros necesarios:

MIGDSN [dsn | *]

Filtro de conjunto de datos - Filtro de conjunto de datos para la lista. El valor predeterminado es *.

MIGMEM [member | *]

Filtro de miembros - Filtro de miembros para la lista. El valor predeterminado es *.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

MIGPDS – Migrar conjuntos de datos y miembros que no son de SCLM en SCLM

Esta función migra los conjuntos de datos y miembros seleccionados que no se gestionan mediante SCLM en SCLM.

```
>>-MIGPDS-_____GROUP-PROJECT-TYPE-><
      | _PROJDEF_ |
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

PROJGRPS – Recuperar grupos SCLM para un proyecto

Esta función recupera los grupos SCLM para un proyecto seleccionado.

```
>>-PROJGRPS-_____PROJECT-><
      | _PROJDEF_ |
```

Parámetros necesarios:

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

PROJINFO – Recuperar información de proyecto SCLM

Esta función recupera información sobre el proyecto SCLM para un proyecto seleccionado.

```
>>-PROJINFO-_____PROJECT-REPDGRP-><
      | _PROJDEF_ |
```


[PRMMSG [Y | N]]

Mensajes de promoción - Establézcalo en Y para incluir mensajes de promoción. El valor predeterminado es N.

[PRMMSGDS [dsn | NONE]]

Conjunto de datos de mensaje de promoción - NONE o el nombre del conjunto de datos que sostendrá los mensajes de promoción. Si el conjunto de datos no existe, se asignará un conjunto de datos nuevo. Los mensajes de promoción también se devolverán en el archivo de respuestas XML, si PRMMSG está establecido en Y. El valor predeterminado es NONE.

[PRMREPT [Y | N]]

Informe de promoción - Establézcalo en Y para incluir informes de promoción. El valor predeterminado es N.

[PRMRPTDS [dsn | NONE]]

Conjunto de datos de informe de promoción - NONE o el nombre del conjunto de datos que sostendrá el informe de promoción. Si el conjunto de datos no existe, se asignará un conjunto de datos nuevo. Los mensajes de promoción también se devolverán en el archivo de respuestas XML, si PRMMSG está establecido en Y. El valor predeterminado es NONE.

[PRMSCOPE [E | N | S]]

Ámbito de promoción - Indica el ámbito de promoción (E=ampliado, N=normal, S=subunidad). El valor predeterminado es N.

[PROJDEF]

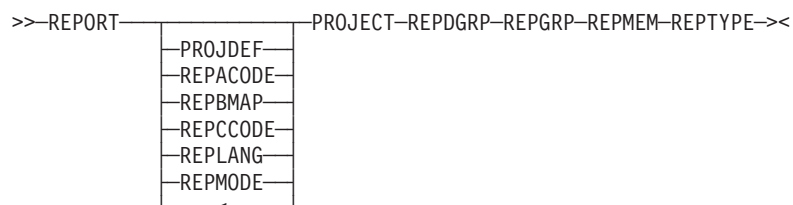
Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[SUBMIT [BATCH | ONLINE]]

Método de sometimiento - La promoción se somete en línea o por lotes. El valor predeterminado es ONLINE.

REPORT – Crear informe de proyecto

La función de informes proporciona un informe de jerarquía DBUTIL del proyecto, de acuerdo con los parámetros y filtros de informe que se utilicen.



Parámetros necesarios:

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

REPGRP {group | group* | * }

Grupo SCLM - Grupo(s) SCLM que se debe(n) filtrar.

REPMEM {member | mem* | * }

Miembro SCLM - Miembro(s) SCLM que se debe(n) filtrar.

REPTYPE {type | type* | * }

Tipo SCLM - Tipo(s) SCLM que se debe(n) filtrar.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[REPACODE]

Código de autoridad - Código de autoridad en el que se debe realizar el filtrado.

[REPBMAP [Y | N]]

Correlaciones de construcción - Distintivo para incluir correlaciones de construcción en el informe DBUTIL. El valor predeterminado es N.

[REPCODE]

Código de cambio - Código de cambio en el que se debe realizar el filtrado.

[REPLANG]

Lenguaje - Tipo de lenguaje SCLM en el que se debe realizar el filtrado.

[REPMODE [D | E]]

Modalidad - Modalidad de desarrollador (D) o explorador (E) con la que se debe realizar el informe. El valor predeterminado es D.

REPUTIL – Informe DBUTIL de SCLM

El servicio DBUTIL recupera información de la base de datos del proyecto y crea una salida personalizada y un informe. Describe el contenido de la base de datos del proyecto basándose en los criterios de selección que suministre.

```
>>--REPUTIL-----GROUP-MEMBER-PROJECT-REPDGRP-TYPE--<
      |
      | PROJDEF
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

SAVE – Guardar miembro SCLM

Esta función copia un miembro del directorio WORKAREA/userid/EDIT/ en SCLM bajo un grupo de desarrollo del usuario. Se creará un miembro nuevo si dicho miembro no existe en la jerarquía de proyectos SCLM.

```
>>--SAVE-----GROUP-MEMBER-PROJECT-TYPE--<
      |
      | PROJDEF
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

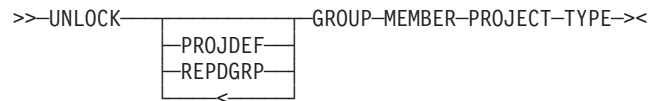
Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

UNLOCK – Desbloquear miembro SCLM

Esta función desbloquea un miembro SCLM que ha sido bloqueado por la función EDIT.



Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

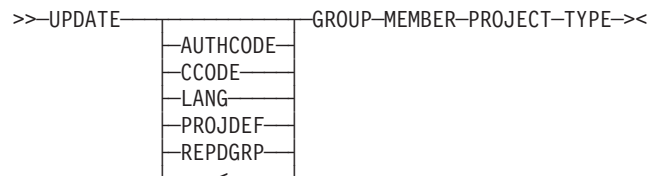
Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[REPDGRP]

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

UPDATE – Actualizar información sobre el miembro SCLM

Esta función actualiza la información sobre el miembro en SCLM.



Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[AUTHCODE]

Código de autorización - Código de autorización con el que se debe actualizar el miembro.

[CCODE]

Código de cambio - Código de cambio con el que se debe actualizar el miembro.

[LANG]

Lenguaje SCLM - El lenguaje SCLM del miembro seleccionado.

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

[REPDGRP]

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

VERBROW – Versiones de navegación de SCLM

Esta función examina una versión de un miembro a partir del conjunto de datos de la versión.

```
>>-VERBROW-PROJDEF-GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

VERDEL – Versiones de supresión de SCLM

Esta función suprime la información acerca de un miembro versionado o auditado de SCLM.

```
>>-VERDEL-PROJDEF-GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
```

Esta función suprime todas las versiones anteriores de un miembro en SCLM.

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

VERHIST – Historial de versiones de SCLM

Esta función muestra el historial de versiones de una versión de miembro seleccionado en SCLM.

```
>>-VERHIST-PROJDEF-GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

VERLIST – Versiones de lista de SCLM

Esta función lista las distintas versiones de un miembro en concreto.

```
>>-VERLIST-PROJDEF-GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

VERREC – Versiones de recuperación de SCLM

Esta función recupera una versión seleccionada de un miembro en un grupo de desarrollo.

```
>>-VERREC-PROJDEF-GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
```

Parámetros necesarios:

GROUP

Grupo SCLM - El grupo SCLM donde reside el miembro seleccionado.

MEMBER

Miembro SCLM - Miembro SCLM seleccionado.

PROJECT

Nombre de proyecto SCLM - El nombre del proyecto SCLM.

REPDGRP

Grupo de desarrollo SCLM - El grupo de desarrollo del usuario.

TYPE Tipo SCLM - El tipo SCLM que contiene el miembro seleccionado.

Parámetros opcionales:

[PROJDEF]

Nombre de proyecto alternativo de SCLM - El nombre de la definición de proyecto (el valor predeterminado es Project).

Muestras

Se suministra un programa Java de muestra (con entrada y salida) que accede a la API de los servicios de host SCLMDT utilizando un servidor HTTP como mecanismo de transporte.

sclmdt_request.xml

Una solicitud de muestra del siguiente ejemplo llama a la función BUILD para compilar y enlazar el miembro ALLOCEXT en el proyecto SCLMVCM.


```

<?xml version="1.0"?>
<SCLMDT-INPUT
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="sclmdt.xsd">
  <SERVICE-REQUEST>
    <service>SCLM</service>
    <session>REUSE</session>
    <bldrept>Y</bldrept>
    <groupbld>DEV1</groupbld>
    <projdef>SCLMVCM</projdef>
    <bldmode>C</bldmode>
    <repdgrp>DEV1</repdgrp>
    <sclmfunc>BUILD</sclmfunc>
    <member>ALLOCEXT</member>
    <project>SCLMVCM</project>
    <group>TEST</group>
    <type>SOURCE</type>
  </SERVICE-REQUEST>
</SCLMDT-INPUT>

```

Figura 24. sclmdt_request.xml – archivo de entrada de mandatos XML de muestra

xmlbld.java

El siguiente ejemplo muestra un programa Java de muestra (en Eclipse) utilizando la solicitud de entrada XML anterior. Cuando se ejecute, establecerá una conexión URL con el servidor HTTP en z/OS (CDFMVS08), pasando la secuencia de entrada XML como una solicitud POST a la rutina CGI BWBXML.

```

package com.ibm.sclmCaller;

import java.io.*;
import java.net.*;
import java.util.*;

public class XMLBLD {

    public static void main(String[] args) {

        try {

            BufferedReader in = new BufferedReader(new FileReader("C:\\logon.txt"));
            String logon = in.readLine();
            in.close();

            Date d = new Date() ;
            System.out.println("START Transfer DATE/TIME is : " + d.toString() );

            // URL details for CGI POST
            URL url = new URL("http", "CDFMVS08", 8080, "/BWBXML");
            HttpURLConnection con = (HttpURLConnection) url.openConnection();

            con.setUseCaches(false);
            con.setDoInput(true);
            con.setDoOutput(true);
            con.setRequestMethod("POST");
            con.setRequestProperty( "Authorization", "Basic "
                + Base64Encoder.encode( logon ));

            System.out.println("At url openConnection.. ");

            // POST CGI routines
            doPut(url, con);
            doGet(url, con);

            Date c = new Date() ;
            System.out.println("TOTAL Completion DATE/TIME is : " + c.toString() );

        }
        catch (IOException exception)
        {
            System.out.println("Error: " + exception);
        }
    }

    public static void doPut(URL url, HttpURLConnection con) throws IOException
    {

        PrintWriter out = new PrintWriter(con.getOutputStream());
        // A continuación se muestra una entrada XML en línea de muestra para
        // una solicitud de servicio ISPF
        // Alternativamente, esto se podría leer desde un archivo externo

        out.println( "<?xml version='1.0'>" );
        out.println( "<SCLMDT-INPUT" );
        out.println( " xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'" );
        out.println( " xsi:noNamespaceSchemaLocation='sclmdt.xsd'" );
        out.println( "<SERVICE-REQUEST>" );
        out.println( "<service>SCLM</service>" );
        out.println( "<session>NONE</session>" );
        out.println( "<ispprof>IBMUSER.TEST.ISPPROF</ispprof>" );
        out.println( "<sclmfunc>BUILD</sclmfunc>" );
        out.println( "<project>SCLMVCM</project>" );
        out.println( "<projdef>SCLMVCM</projdef>" );
        out.println( "<member>ALLOCEXT</member>" );
        out.println( "<group>TEST</group>" );
        out.println( "<type>SOURCE</type>" );
        out.println( "<repdgrp>DEV1</repdgrp>" );
        out.println( "<groupbld>DEV1</groupbld>" );
        out.println( "<bldrept>Y</bldrept>" );
        out.println( "<bldmsg>Y</bldmsg>" );
        out.println( "<bldmode>C</bldmode>" );
    }
}

```

1. Dependiente de Base64Encoder.class para el cifrado de ID de usuario.
2. El archivo C:\logon.txt contiene USERID:PASSWORD.

El ejemplo siguiente muestra la salida de muestra de la función BUILD invocada utilizando el programa Java de muestra.

Figura 26. sclmdt_response.xml – archivo de salida XML de muestra

Apéndice C. API de SCLM Developer Toolkit 89

```

**                ALTERNATE:  SCLMVCN                **
**                SCOPE:      NORMAL                  **
**                MODE:       CONDITIONAL              **
**                                                    **
*****
*****
1  *** B U I L D   O U T P U T S   G E N E R A T E D *** Page 1

MEMBER      TYPE      VERSION      KEYWORD
-----
***** NO MODULES GENERATED *****
1  ***** B U I L D   M A P S   G E N E R A T E D ***** Page 2

MEMBER      TYPE      VERSION      (REASON FOR REBUILD)
-----
***** NO BUILD MAPS GENERATED *****
1  ***** B U I L D   O U T P U T S   D E L E T E D ***** Page 3

MEMBER      TYPE      VERSION      KEYWORD
-----
***** NO MODULES DELETED *****
1  ***** B U I L D   M A P S   D E L E T E D ***** Page 4

MEMBER      TYPE      VERSION      (REASON FOR DELETE)
-----
***** NO BUILD MAPS DELETED *****
]]&#62;
</BUILD-REPORT>
<SCLM-MESSAGES>
  <SCLM-MESSAGE ID="FLM87107">- BUILD SUCCEEDED FOR MEMBER ALLOCEXT AT
    09:53:01, CODE: 0</SCLM-MESSAGE>
</SCLM-MESSAGES>
</SERVICE-RESPONSE>
<OPERATIONS-LOG>
<![CDATA[
  Status: 200
  Content-Type: text/plain

Entering BWBINT (Service initialization)
Host driver level : V4 12Feb08
09:52:57.48
Function ID timestamp = ID35577
Environment variables :
1 CONTENT_TYPE application/x-www-form-urlencoded
2 QUERY_STRING
3 PATH /usr/lpp/rdz/bin:/bin:./usr/sbin:/usr/lpp/internet/bin:/usr/lpp/
  internet/sbin:/usr/lpp/java/J5.0/bin
4 AUTH_TYPE Basic
5 DOCUMENT_URI /BWFXML
6 SHELL /bin/sh
7 HTTPS OFF
8 HTTP_USER_AGENT Java/1.5.0
9 HTTP_ACCEPT text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
10 RULE_FILE //DD:CONF
11 SERVER_PORT 8080
12 CONTENT_LENGTH 554
13 PATH_INFO
14 GATEWAY_INTERFACE CGI/1.1
15 _CEE_RUNOPTS ENVAR("_CEE_ENVFILE=//DD:ENV")
16 REFERER_URL
17 _BPX_SPAWN_SCRIPT YES
18 _ ./BWBCRON1
19 CLASSPATH ./usr/lpp/internet/server_root/CAServlet

```

```

20 REMOTE_ADDR 192.168.124.236
21 REQUEST_METHOD POST
22 STEPLIB CURRENT
23 CGI_TRANTABLE FEK.#CUST.LSTRANS.FILE
24 LANG C
25 REMOTE_USER IBMUSER
26 LIBPATH /bin:/usr/lpp/internet/bin:/usr/lpp/internet/sbin
27 FSCP IBM-1047
28 SERVER_ADDR 56. 9.42.112.75
29 HTTP_CONNECTION keep-alive
30 PATH_TRANSLATED
31 HTTP_HOST CFDFMVS08
32 SERVER_TOKEN 1
33 HTTP_CACHE_CONTROL no-cache
34 SERVER_SOFTWARE IBM HTTP Server/V5R3M0
35 _BPX_SHAREAS YES
36 NETCP ISO8859-1
37 DOCUMENT_ROOT /usr/lpp/rdz/bin
38 REPORTBITS 77
39 COUNTERDIR NULL
40 LC_ALL en_US.IBM-1047
41 SERVER_PROTOCOL HTTP/1.1
42 _BPX_USERID IBMUSER
43 _SCLMDT_CONF_HOME /etc/rdz/scldmt
44 HTTPS_KEYSIZE
45 JAVA_HOME /usr/lpp/java/J5.0/
46 TZ EST5EDT
47 SCRIPT_NAME /BWBXML
48 _BPX_BATCH_SPAWN SPAWN
49 _CEE_ENVFILE //DD:ENV
50 NLSPATH /usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:/usr/
lib/nls/msg/En_US.IBM-1047/%N
51 DOCUMENT_NAME /usr/lpp/rdz/bin/BWBXML
52 _SCLMDT_WORK_HOME /var/rdz
53 HTTP_PRAGMA no-cache
54 SERVER_NAME CFDFMVS08
Timecheck1:09:52:57.49
Connection Protocol : HTTP
Server Name : CFDFMVS08
Server Port : 8080
SCRT check: /var/rdz/WORKAREA/scrtTimeStamp Time:1206492777
File: 1206492602
Timecheck2:09:52:57.51
Timecheck2b:09:52:57.51
FSCP = IBM-1047
NETCP = ISO8859-1
_SCLMDT_CONF_HOME = /etc/rdz/scldmt
_SCLMDT_WORK_HOME = /var/rdz
CGI_TRANTABLE = FEK.#CUST.LSTRANS.FILE
Server PATH = /usr/lpp/rdz/bin:/bin:/usr/sbin:/usr/lpp/internet/
bin:/usr/lpp/internet/sbin:/usr/lpp/java/J5.0/bin

Check: userdir = /var/rdz/WORKAREA/IBMUSER
Timecheck1sa:09:52:57.51
Timecheck3:09:52:57.51
** Development Group = DEV1
** Selected Group = TEST
Plugin version : NONE
Host version : 4.1.0
Temporary data set prefix set to : SCLMVCM.IBMUSER

Timecheck4:09:52:58.04
Parameters to be written to temporary data set 'SCLMVCM.IBMUSER.SCLMDT.
VCMISPF.ID35577' : ISPPROF=IBMUSER.TEST.ISPPROF&SCLMFUNC=BUILD
&PROJECT=SCLMVCM&PROJDEF=SCLMVCM&MEMBER=ALLOEXT&GROUP=TEST
&TYPE=SOURCE&REPDGRP=DEV1&GROUPBLD=DEV1&BLDREPT=Y&BLMSG=Y&BLDMODE=C
/etc/rdz/scldmt;/var/rdz

```

```
/usr/lpp/rdz/bin:/bin:./usr/sbin:/usr/lpp/internet/bin:/usr/lpp/internet/
sbin:/usr/lpp/java/J5.0/bin/~~FEK.#CUST.LSTRANS.FILE
```

```
Processing SCLM request :
Value for SESSFLG = NONE
Value for SESSION = NONE
Value for SCLMFUNC = BUILD
Value for PROJ = SCLMVCM
Value for PROJDEF = SCLMVCM
Value for Development GROUP = DEV1
Value for Selected GROUP = TEST
Value for TYPE = SOURCE
Value for LANG = NONE
Value for MEMBER = ALLOEXT
Value for J2EEFILE = NONE
Value for PROFDSN = IBMUSER.TEST.ISPPROF
Value for PARMS = NONE
pcnt = 3
parmline.1 = ISPF SELECT CMD(BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM SCLMVCM
TEST DEV1 SOURCE ALLOEXT /) NEST LANG(CREX)
time check before ISPZINT call :09:52:58.17
time check after ISPZINT call :09:53:02.51
ispzint rc = 0
check: respline count = 226
*** ISPZINT OUTPUT ***
Content-type: text/plain
```

```
Entering ISPZINT (Service initialization)
About to read from fileno(stdin) = 0
Data read from STDIN is ISPF SELECT CMD(BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM
SCLMVCM TEST DEV1 SOURCE ALLOEXT /) NEST LANG(CREX)
EPOCH secs = 1206492778
Local Date & time: Tue Mar 25 20:52:58 2008
Hour: 20 Min: 52 Sec 58
Function ID timestamp = ID075178
Environment variables:
0 QUERY_STRING=
1 CONTENT_TYPE=application/x-www-form-urlencoded
2 PATH=/usr/lpp/rdz/bin:/bin:./usr/sbin:/usr/lpp/internet/bin:/usr/lpp/
internet/sbin:/usr/lpp/java/J5.0/bin
3 AUTH_TYPE=Basic
4 DOCUMENT_URI=/BWBXML
5 SHELL=/bin/sh
6 HTTPS=OFF
7 HTTP_ACCEPT=text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
8 HTTP_USER_AGENT=Java/1.5.0
9 SERVER_PORT=8080
10 RULE_FILE=/DD:CONF
11 GATEWAY_INTERFACE=CGI/1.1
12 PATH_INFO=
13 CONTENT_LENGTH=554
14 _CEE_RUNOPTS=ENVAR(" _CEE_ENVFILE=/DD:ENV")
15 _BPX_SPAWN_SCRIPT=YES
16 REFERER_URL=
17 _=/u/SCLMDTIS/bin/ISPZINT
18 CLASSPATH=./usr/lpp/internet/server_root/CAServlet
19 STEPLIB=CURRENT
20 REQUEST_METHOD=POST
21 REMOTE_ADDR=192.168.128.236
22 CGI_TRANTABLE=FEK.#CUST.LSTRANS.FILE
23 LANG=C
24 LIBPATH=/bin:/usr/lpp/internet/bin:/usr/lpp/internet/sbin
25 REMOTE_USER=IBMUSER
26 SERVER_ADDR=9.42.112.75
27 FSCP=IBM-1047
28 PATH_TRANSLATED=
29 HTTP_CONNECTION=keep-alive
```

```

30 SERVER_TOKEN=1
31 HTTP_HOST=CDFMVS08
32 _BPX_SHAREAS=YES
33 SERVER_SOFTWARE=IBM HTTP Server/V5R3M0
34 HTTP_CACHE_CONTROL=no-cache
35 REPORTBITS=77
36 DOCUMENT_ROOT=/usr/lpp/rdz/bin
37 NETCP=ISO8859-1
38 COUNTERDIR=NULL
39 LC_ALL=en_US.IBM-1047
40 _SCLMDT_CONF_HOME=/etc/rdz/sclmdt
41 _BPX_USERID=IBMUSER
42 SERVER_PROTOCOL=HTTP/1.1
43 JAVA_HOME=/usr/lpp/java/J5.0/
44 HTTPS_KEYSIZE=
45 TZ=EST5EDT
46 _CEE_ENVFILE=//DD:ENV
47 _BPX_BATCH_SPAWN=SPAWN
48 SCRIPT_NAME=/BWBXML
49 NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:
/usr/lib/nls/msg/En_US.IBM-1047/%N
50 _SCLMDT_WORK_HOME=/var/rdz
51 DOCUMENT_NAME=/usr/lpp/rdz/bin/BWBXML
52 SERVER_NAME=CDFMVS08
53 HTTP_PRAGMA=no-cache
Number of environment variables is 54
Connection Protocol = HTTP
Server Name = CDFMVS08
Server Port = 8080
***ERROR: Unable to get status information for ISPZTS0
FSCP = IBM-1047
NETCP = ISO8859-1
Server PATH = /usr/lpp/rdz/bin:/bin:/usr/sbin:/usr/lpp/internet/bin:/
usr/lpp/internet/sbin:/usr/lpp/java/J5.0/bin/
ISPF standalone function invoked
ISPF COMMAND = ISPF SELECT CMD(BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM
SCLMVCM TEST DEV1 SOURCE ALLOCEXT /) NEST LANG(CREX)
ISPF PROFILE = NONE
Re-usable ISPF session =
About to spawn task for ISPZTS0
Parameters passed to ISPZTS0 - PROFILE
Return code from ISPZTS0 is 0
About to process PROFILE data in /tmp/IBMUSER.ID075178.ISPF.SYSTSPRT
About to malloc() 252 bytes for profdat
Temporary data set prefix set to : SCLMVCM
About to call bpxwdyn to allocate VCMTMP
Allocating data set SCLMVCM.ISPF.VCMISPF.ID075178 to the VCMTMP DD
1024 bytes of ISPF SELECT CMD(BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM
SCLMVCM TEST DEV1 SOURCE ALLOCEXT /) NEST LANG(CREX)
written to VCMTMP
1024 bytes of /etc/rdz/var/rdz written to VCMTMP
1024 bytes of /usr/lpp/rdz/bin:/bin:/usr/sbin:/usr/lpp/internet/bin:
/usr/lpp/internet/sbin:/usr/lpp/java/J5.0/bin/~~
written to VCMTMP
Parameter to be passed to ISPZTS0 CALL *(ISPZCNT) 'ISPF ID075178 SCLMVCM
NONE ISPF SELECT CMD(BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM
SCLMVCM TEST DEV1 SOURCE ALLOCEXT /) NEST LANG(CREX)'
About to spawn task for ISPZTS0
Parameters passed to ISPZTS0 - CALL *(ISPZCNT) 'ISPF ID075178 SCLMVCM
NONE ISPF SELECT CMD(BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM
SCLMVCM TEST DEV1 SOURCE ALLOCEXT /) NEST LANG(CREX)'
Return code from ISPZTS0 is 0
About to open /tmp/IBMUSER.ID075178.ISPF.SYSTSPRT
IKJ56003I PARM FIELD TRUNCATED TO 100 CHARACTERS
Entering ISPZCNT (ISPF Initialization)
Parameters ISPF ID075178 SCLMVCM NONE ISPF SELECT CMD(BWBBLD ID35577
SCLMVCM.IBMUSER SCLMVCM SCLMVCM TEST DEV1

```

```

REC: isplib=isp.sispmenu
Allocation successful for ISPLIB
REC: isptlib=isp.sisptenu
Allocation successful for ISPTLIB
REC: ispllib=isp.sisppenu
Allocation successful for ISPLLIB
REC: ispslib=bzz.sbzzenus,isp.sispsenu,isp.sispslib
Allocation successful for ISPSLIB
REC: ISPF_timeout = 300
NOTE: Data set allocations took 0.28 elapsed seconds
Running user ISPF command: ISPSTART CMD(BWBBLD ID35577 SCLMVCM.IBMUSER
      SCLMVCM SCLMVCM TEST DEV1 SOURCE ALLOCEXT /) NEST
<ISPINFO>
ISPF COMMAND : ISPSTART CMD(BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM
      SCLMVCM TEST DEV1 SOURCE ALLOCEXT /) NEST LANG(CREX) N
<ISPF>
Entering BWBBLD
Temporary data set prefix : SCLMVCM.IBMUSER
about to allocate VCMISPF
rc from alloc is 0
Translate table allocated successfully
SCLMDT CONFIG directory = /etc/rdz/scldmt
SCLMDT Working directory = /var/rdz
SCLMDT Translate table = FEK.#CUST.LSTRANS.FILE

***** BUILD PARMS *****
PROJECT = SCLMVCM
PROJDEF (Project Name) = SCLMVCM
GROUP (SCLM Group) = TEST
GROUPBLD (Build at group) = DEV1
TYPE (SCLM Type) = SOURCE
MEMBER (SCLM Build Member) = ALLOCEXT
BLDScope (Build Scope) = N
BLDMODE (Build Mode) = C
BLDLIST (Listing Flag ) = Y
BLDREPT (Report Flag ) = Y
BLDMSG (Messages Flag ) = Y
BLDLSTDS (Listing Data set name) =
BLDRPTDS (Report Data set name) =
BLDMSGDS (Messages Data set name) =
BLDEXTDS (Exit Data set name) =
SUBMIT (Submission type) = ONLINE
***** End PARMS *****

projfile = /etc/rdz/scldmt/CONFIG/PROJECT/SCLMVCM.conf
Security Flag set to N
rc from FLMSGSGS alloc is 0
PARMS=SCLMVCM,SCLMVCM,DEV1,SOURCE,ALLOCEXT,,N,C,Y,Y,,tmpmsg,
      tmprpt,tmp1st,BLDEXIT
BWBBLD CHECK: PARMS = SCLMVCM,SCLMVCM,DEV1,SOURCE,ALLOCEXT,,
      N,C,Y,Y,,tmpmsg,tmp1st,tmp1st,BLDEXIT
*** BUILD SUCCESSFUL ***

Cleaning up workarea directories

***** SCLM MESSAGES *****
FLM87107 - BUILD SUCCEEDED FOR MEMBER ALLOCEXT AT 09:53:01, CODE: 0
*****

*** XML-NOTE *** Reference tagged SERVICE-RESPONSE
</ISPF>
RC=0
</ISPINFO>
ISPRC = 0
***** ISPLLOG CONTENTS *****
1 Time *** ISPF transaction log *** Userid: IBMUSER Date: 08/03/26 Page:

```



```

09:53 Start of ISPF Log - - - Session # 1 -----
09:53 TSO - Command - - BWBBLD ID35577 SCLMVCM.IBMUSER SCLMVCM
      SCLMVCM TEST DEV1 SOURCE ALLOCEXT /
09:53 TSO - Command - - FLMCMD
09:53 BUILD,SCLMVCM,SCLMVCM,DEV1,SOURCE,ALLOCEXT
      ,,N,C,Y,Y,,tmpmsg,tmpcpt,tmp1st,BLD
09:53 EXIT
09:53 End of ISPF Log - - - Session # 1 -----
***** END ISPLLOG CONTENTS *****
IKJ56247I FILE ISPLLIB NOT FREED, IS NOT ALLOCATED
Leaving ISPZCNT
READY
END
return code from ISPFInit = 0
PROCESSING COMPLETE
End of SCLM Processing
FUNC: BUILD - Cleaning up old 'SCLMVCM.IBMUSER.SCLMDT.VCMISPF.ID35577'
EXIT BWBINT 09:53:02.86 WITH RC=0
]]&#62;
</OPERATIONS-LOG>
</SCLMDT-OUTPUT>
TOTAL Completion DATE/TIME is :Wed Mar 26 09:44:11 WST 2008

```

Apéndice D. Programa de utilidad de construcción de Rational Application Developer for WebSphere Software

Visión general del Programa de utilidad de construcción de Rational Application Developer for WebSphere Software

Esta sección aborda las mejoras conseguidas en el proceso de construcción Java/J2EE mediante el Programa de utilidad de construcción de Rational Application Developer for WebSphere Software (anteriormente conocido como AST: Application Server Toolkit). El Programa de utilidad de construcción de Rational Application Developer for WebSphere Software se proporciona como parte de Rational Application Developer (RAD). RAD debe pedirse, instalarse y configurarse aparte. La instalación y la personalización de este producto no se describen en este manual. Consulte la documentación proporcionada con RAD para obtener instrucciones de instalación y personalización para la función de Programa de utilidad de construcción. En esta sección se hará referencia al Programa de utilidad de construcción de Rational Application Developer for WebSphere como "el Programa de utilidad de construcción".

El Programa de utilidad de construcción permite espacios de trabajo de proyecto replicados de IDE que se hayan almacenado en SCLM para que los construya SCLM utilizando la modalidad autónoma de Eclipse en z/OS.

Esta sección debe utilizarse junto con la Guía de usuario en línea de SCLM Developer Toolkit así como con la Guía de instalación y de personalización que se suministra con el producto. SCLM Developer Toolkit proporciona conversores de lenguaje Java/J2EE para SCLM a fin de habilitar todas las prestaciones de construcción de Java/J2EE.

SCLM Developer Toolkit facilita la funcionalidad para almacenar objetos y fuentes de JAVA/J2EE y, además, mediante estos conversores permite que SCLM construya y controle completamente las aplicaciones de Java/J2EE. Los objetos de Java/J2EE soportados por estos conversores de lenguaje son clases Java, archivadores Java (JAR), Enterprise JavaBeans (archivos JAR EJB), archivadores web (WAR) y archivadores empresariales (EAR).

SCLM Developer Toolkit se integra con el Programa de utilidad de construcción para controlar proyectos SCLM J2EE que se replican en el espacio de trabajo del Programa de Utilidad de construcción bajo UNIX System Services on z/OS. A continuación, el proceso de construcción ARCHDEF construye los proyectos duplicados que hacen referencia a los scripts del Programa de utilidad de construcción. Se entregan scripts de construcción de muestra con SCLM Developer Toolkit. Los objetos resultantes de la construcción/compilación se vuelven a almacenar en SCLM.

El Programa de utilidad de construcción soporta construcciones de todos los proyectos J2EE soportados en Rational Application Developer V7. Los proyectos contruidos en versiones anteriores de RAD deberán migrarse en el espacio de trabajo RAD V7 de IDE Client antes de añadirlos a SCLM. También se soportan los proyectos Java de Eclipse normales.

Almacenamiento de objetos Java/J2EE en SCLM

Los objetos Java/J2EE se almacenan en SCLM de la siguiente manera:

- La fuente Java se compila en clases. Las clases se almacenan en SCLM en el tipo JAVACLAS. El nombre corto y el nombre largo se almacenan en la tabla de conversión.
- Soporte para EJB y archivos JAR normales (contiene clases y es posible que contenga otros componentes de proyecto Java tales como XML/HTML/JSP en una estructura empaquetada). Archivos JAR almacenados en SCLM tipo J2EEJAR.
- Soporte para archivos WAR ensamblados basados en el archivo J2EE web.xml en el proyecto J2EE. Archivos WAR almacenados en SCLM tipo J2EEWAR.
- Soporte para archivos EAR creados para el despliegue basado en application.xml en el proyecto J2EE. Archivos EAR almacenados en SCLM tipo J2EEEAR.
- Soporte al despliegue de archivos EAR en Websphere Application Server (WAS) en z/OS.
- Soporte para la construcción SQLJ
- Todos los listados de salida se almacenan en SCLM tipo J2EELIST.

El Programa de utilidad de construcción comparado con el proceso de construcción ANT nativo

SCLM Developer Toolkit suministra un proceso de construcción ANT nativo para el soporte de Java/J2EE. Consulte el conector de la Guía del usuario de SCLM Developer Toolkit en línea y la Guía de instalación/personalización.

El Programa de utilidad de construcción de Rational Application Developer for WebSphere Software amplía el soporte de construcción de Java/J2EE duplicando completamente el entorno IDE de Eclipse en z/OS durante la construcción.

Como en el proceso de construcción ANT, el proceso de Programa de utilidad de construcción utiliza la archdef, que contiene miembros que constituyen el proyecto Java/J2EE y son una representación con nombre corto del modo en que el proyecto existe en un espacio de trabajo de Eclipse.

El usuario puede elegir el proceso de Programa de utilidad de construcción o el proceso de construcción ANT nativo asignando el script de construcción (al que la ARCHDEF hace referencia) con el conversor de lenguaje adecuado:

- J2EEAST para el Programa de utilidad de construcción
- J2EEANT para la construcción ANT nativa

Cuando se construye la archdef, invoca al conversor de lenguaje de verificación previo a la construcción (J2EEAST). Este tipo de lenguaje J2EEAST se asigna al script de construcción al que la palabra clave SING hace referencia en la archdef. Este conversor de lenguaje copia todos los componentes archdef en el sistema de archivos de z/OS UNIX System Services que debe construirse mediante el Programa de utilidad de construcción.

Los componentes del proyecto y la fuente Java se pueden almacenar en SCLM como EBCDIC o ASCII. En el Programa de utilidad de construcción, los componentes de texto y la fuente Java se convierten de forma predeterminada en ASCII en el espacio de trabajo USS antes de que se produzca el proceso de construcción.

Notas de instalación del Programa de utilidad de construcción de Rational Application Developer for WebSphere Software

El Programa de utilidad de construcción de Rational Application Developer for WebSphere Software se proporciona como parte de Rational Application Developer (RAD). RAD debe pedirse, instalarse y configurarse aparte. La instalación y la personalización de este producto no se describen en este manual. Consulte la documentación proporcionada con RAD para obtener instrucciones de instalación y personalización para la función de Programa de utilidad de construcción. El paquete del directorio del Programa de utilidad de construcción deberá estar ubicado en el sistema de archivos de z/OS UNIX.

Es posible que el proceso de construcción J2EE necesite tamaños de región grandes para evitar quedarse sin memoria o que surjan errores de almacenamiento. Para conseguir construcciones en línea grandes, especifique un valor mínimo REGION=512M en el daemon RSE de System z (esta es la tarea iniciada RSED predeterminada). Para el proceso por lotes, es recomendable haber especificado este parámetro de tamaño de región en el lote JOBCARD.

Integración de SCLM con el Programa de utilidad de construcción

El proceso J2EEAST inicial es parecido a J2EEANT en el que el conversor de lenguaje determina, a partir de la archdef, qué partes se deben volver a construir en función de la modalidad de construcción (condicional o forzada). A continuación, se copian los archivos fuente de proyecto y los objetos generados incluidos en el área de trabajo del sistema de archivos de UNIX Systems services para el Programa de utilidad de construcción. El script de ejecución y el XML de construcción, que se almacenan en SCLM en el tipo J2EEBLD, también se copian en la misma área de trabajo. Se invoca al script de ejecución para que Eclipse aparezca en modalidad autónoma en z/OS y para ejecutar el XML de construcción de aplicación referenciado. El Programa de utilidad de construcción compila y genera los objetos Java/J2EE necesarios, especificados por el script de ejecución, el XML de construcción y la archdef.

J2EEAST examina los objetos generados y sólo selecciona aquellas partes/objetos para la actualización SCLM, que debían volver a construirse según SCLM y desde la modalidad de construcción (condicional/forzada).

SCLM procesa cada componente archdef individual que ejecute cada conversor de lenguaje asociado con dicho componente. El conversor de lenguaje JAVA asociado con cada miembro fuente Java copia de nuevo los archivos de clase asociados con cada uno de ellos en SCLM.

La parte final del proceso de construcción es el proceso de la propia ARCHDEF, en la que se ejecuta el conversor de lenguaje J2EEOBJ. Este conversor archdef determina qué objetos J2EE se han generado (JAR, WAR, EAR) y vuelve a copiar estas partes en SCLM.

Implementación y utilización del Programa de utilidad de construcción de Rational Application Developer for WebSphere Software

La implementación y la utilización se efectúan de la siguiente manera:

- De forma predeterminada, SCLM Developer Toolkit se suministra con asistentes de archdef y de script de construcción que generan archdefs de proyecto y scripts de construcción asociados. Éstos generan scripts de proceso de

construcción ANT nativos y asignan los scripts de construcción a un lenguaje de J2EEANT. SCLM Developer Toolkit proporciona un recuadro de selección en la página Preferencias de construcción de SCLM para permitir que estos asistentes generen en su lugar scripts de construcción.

- La página **Equipo > Preferencias de SCLM > Opciones de script de construcción** debería tener cada muestra de script de construcción (Java, EJB, WAR, EAR, SQLJ) asociada con un script de ejecución de BWBASTR.
- El script de ejecución (muestra BWBASTR) deberá copiarse en cada proyecto SCLM que requiera el Programa de utilidad de construcción. Dicho script deberá residir en SCL, tipo J2EEBLD, y deberá ser del tipo de lenguaje TEXT o J2EEPART. Este script debe personalizarse. Dentro del propio script de ejecución encontrará instrucciones acerca de cómo personalizarlo. Consulte los formatos y scripts del Programa de utilidad de construcción para obtener más información acerca del script de ejecución BWBASTR.
- Cada componente de proyecto (JAR, WAR, EAR) necesita tanto una archdef (consulte los scripts del Programa de utilidad de construcción para obtener información sobre el diseño del formato de la archdef) así como un script de construcción correspondiente (consulte la sección 2.4 para obtener información sobre diseños de script de construcción). Estos scripts de construcción deben crearse en un tipo SCLM de J2EEBLD y deben definirse con un tipo de lenguaje de J2EEST. El usuario puede crear estos scripts de construcción y las archdefs en SCLM, o el usuario también puede generarlos utilizando los asistentes de script de construcción y el de la archdef de cliente, mediante **EQUIPO > Generar script de construcción Java/J2EE** o **EQUIPO > Añadir a SCLM**.
- El sistema devolverá detalles sobre la anomalía en la compilación o la construcción del componente en el registro de operaciones de construcción. Si falla una construcción, puede localizar los errores de compilación buscando el "LISTADO DE COMPILACIÓN" en el registro de operaciones. Si el registro indica un error relacionado con la configuración o la instalación del Programa de utilidad de construcción, póngase en contacto con el administrador del sistema. Los registros de anomalías de construcción de Eclipse se pueden encontrar en el sistema de archivos de UNIX System Services en el directorio `AST_INSTALL/Eclipse/configuration`. Estos registros de error específicos no se podrán leer en z/OS puesto que tendrán un formato ASCII. Deberán convertirse en EBCDIC o transferirse en modalidad binaria al escritorio de Windows® para su examen.
- Si ejecuta construcciones J2EE en modalidad de proceso por lotes, es recomendable incluir el parámetro de tamaño de región `REGION=512M` en la tarjeta de trabajo por lotes. Es posible que los tamaños de región más pequeños produzcan problemas de almacenamiento del tipo 'sin memoria'.

Conversores de lenguaje de Programa de utilidad de construcción de SCLM

Para el Programa de utilidad de construcción se asignan determinados conversores de lenguaje a la fuente J2EE, a la ARCHDEF y al script de construcción.

Se suministran muestras para generar estos lenguajes dentro de un proyecto SCLM con SCLM Developer Toolkit en la biblioteca de muestra de instalación SBWBSAMP.

Código fuente Java

Lenguaje = JAVA | JAVABIN

Texto J2EE

Lenguaje = J2EEPART | J2EEBIN (por ejemplo, XML)

J2EE binario

Lenguaje = J2EEBIN (por ejemplo, jpg)

script de construcción

Lenguaje = J2EEAST

script de ejecución

Lenguaje = TEXT | J2EEPART

ARCHDEF

Lenguaje = archdef

Nota: La palabra clave LKED=J2EEOBJ procesará la archdef con el conversor de lenguaje J2EEOBJ.

J2EEAST: Conversor de verificación/construcción de J2EE

Se trata de un conversor de lenguaje de XML de script de Programa de utilidad de construcción.

Resumen: MUESTRA: BWBTRAN4

El conversor de verificación J2EEAST se invoca cuando se construye la archdef. J2EEAST es el tipo de lenguaje del XML de construcción de J2EEBLD al que la palabra clave SINC hace referencia en la archdef. Este conversor de verificación copia la fuente seleccionada y todos los objetos de la archdef independientemente de la modalidad de construcción del espacio de trabajo de HFS de z/OS que debe referenciar durante el tiempo de construcción como el espacio de trabajo del proyecto. Este espacio de trabajo del proyecto existirá temporalmente en el área de trabajo de SCLM Developer Toolkit. SCLM DT crea un espacio de trabajo del proyecto temporal en la siguiente estructura de directorio: /var/SCLMDT/WORKAREA/userid/project/group/type/member/project

El Programa de utilidad de construcción necesita toda la estructura de proyecto, por lo que todas las partes del proyecto se copian en el sistema de archivos de UNIX System Services. Todos los componentes basados en texto se copian en el espacio de trabajo del proyecto como ASCII. El XML de construcción también se copia en el área de trabajo HFS. El XML de construcción contiene detalles del proyecto, además de otros detalles de propiedades a los que SCLM hace referencia.

El XML de construcción hace referencia a un script de ejecución de Programa de utilidad de construcción asociado. Este script de ejecución también se copia en el área de trabajo (EBCDIC) y cuando se ejecuta, presenta Eclipse en modalidad autónoma en z/OS para ejecutar el XML de construcción de aplicación. La implementación actual redundará en una construcción completa en el espacio de trabajo, a pesar de que SCLM sólo procesará las partes seleccionadas, si se ha solicitado la construcción condicional y el conversor determina los objetos generados que deben actualizarse en SCLM de acuerdo con la modalidad de construcción.

Los objetos J2EE generados tales como archivos de archivado JAR, WAR o EAR se volverán a almacenar en SCLM cuando se ejecute el conversor de archdef J2EEOBJ. Los archivos de clase seleccionados se registran y se actualizan en SCLM, cuando la archdef procesa componentes de Java individuales (conversor Java).

J2EEOBJ: Conversor ARCHDEF de J2EE

Se trata de un conversor de lenguaje de archdef al que la palabra clave LKED hace referencia.

Resumen: MUESTRA: BWBTRAN3

Es el conversor de construcción final que se invoca como parte del proceso de construcción de archdef. Este conversor determina qué objetos J2EE se construyeron previamente en el conversor J2EEAST y los copia en SCLM con el nombre corto generado que se ha suministrado.

1. Si se ha establecido la variable SCLM_BLDMAP, recupere la información de correlación de construcción y actualice el objeto J2EE en el directorio META-INF.
2. Copie estos objetos J2EE (JAR, WAR, EAR) en SCLM con el nombre corto asociado.
 - JAR/EJB en SCLM tipo J2EEJAR
 - WAR en SCLM tipo J2EEWAR
 - EAR en SCLM tipo J2EEEAR
3. Copie los archivos detallados del registro de construcción en SCLM tipo J2EELIST

JAVA: Conversor de lenguaje Java (EBCDIC)

Se trata de un conversor de lenguaje de fuente Java almacenado en EBCDIC.

Resumen: MUESTRA: BWBTRAN1

Tipo de lenguaje para la fuente Java definido por BWBTRAN1 de muestra. El conversor de Java determina qué tipo de construcción se ha emitido contra la fuente Java. Nota: Esta definición de lenguaje debe asignarse a programas Java si desea almacenar la fuente Java en EBCDIC en el host (es decir, es posible visualizar y editar directamente la fuente en el host a través de ISPF). La ventaja de definir programas con esta definición de lenguaje es la posibilidad de poder editar y visualizar la fuente directamente en el host de z/OS. Las desventajas son que las conversiones de página de códigos deben realizarse al migrar o importar proyectos desde el cliente al host.

JAVABIN: Conversor de lenguaje Java (ASCII)

Se trata de un conversor de lenguaje de fuente Java almacenado en ASCII.

Resumen: MUESTRA: BWBTRAN1

Tipo de lenguaje parecido a Java que se utiliza al almacenar la fuente Java como ASCII en SCLM.

J2EEPART: Conversor de lenguaje de texto J2EE

Se trata de un conversor de lenguaje de componente de texto J2EE almacenado en EBCDIC.

Resumen: MUESTRA: BWBTRANJ

J2EEPART es un tipo de lenguaje que especifica un componente JAVA/J2EE y está definido por BWBTRANJ de muestra. No se produce ningún análisis en particular al construir esta definición de lenguaje. Los componentes que no son de fuente Java ni J2EE que necesitan conversión de lenguaje ASCII/EBCDIC pueden insertarse genéricamente bajo esta definición de lenguaje si no se necesita ningún análisis de construcción en particular (por ejemplo, HTML, XML, .classpath, .project, tablas de definición). Se puede utilizar una definición de lenguaje opcional de TEXT.

J2EEBIN: Conversor de lenguaje J2EE Binary

Se trata de un conversor de lenguaje de fuente Java almacenado en EBCDIC.

Resumen: MUESTRA: BWBTRANJ

Tipo de lenguaje que especifica un componente almacenado JAVA/J2EE binario o ASCII y está definido por BWBTRANJ de muestra. No se produce ningún análisis en particular al construir esta definición de lenguaje. Los archivos binarios JAVA/J2EE y los archivos de texto que desee que se almacenen como ASCII pueden insertarse genéricamente bajo esta definición de lenguaje si no se necesita ningún análisis de construcción en particular.

Scripts y formatos de construcción del Programa de utilidad de construcción

Como con el método de servicio de construcción J2EE tradicional, el método de Programa de utilidad de construcción implica que un script de Programa de utilidad de construcción sea referenciado por la ARCHDEF, por la palabra clave SINC.

El script de construcción invocado es una aplicación de Programa de utilidad de construcción para construir XML que debería personalizarse y ser exclusivo para cada proyecto J2EE. Este script de construcción también hace referencia a un script RUN de Programa de utilidad de construcción que contiene propiedades de Programa de utilidad de construcción globales y mandatos de tiempo de ejecución Eclipse para activar Eclipse en z/OS en modalidad autónoma y ejecutar el XML de construcción seleccionado. Normalmente, el script BWBASTR personalizado será el que utilicen todos los scripts de construcción de Programa de utilidad de construcción. El tipo de lenguaje para todos los scripts de construcción de Programa de utilidad de construcción debe ser J2EEAST. El tipo de lenguaje para el script BWBASTR debería ser un conversor de lenguaje de sólo texto como TEXT o J2EEPART.

El script de ejecución de Programa de utilidad de construcción (BWBASTR) , los scripts de construcción de muestra para proyectos Java/Jar (BWBASTJ), el proyecto EJB (BWBASTEJ), el proyecto de cliente de aplicaciones (BWBASTAP), el proyecto web (BWBASTW) y el proyecto EAR (BWBASTE) pueden copiarse desde la biblioteca de muestra de Programa de utilidad de construcción de SCLM Developer Toolkit en los proyectos SCLM de los clientes en el tipo J2EEBLD.

Formato de script de construcción

El formato de script de construcción incluye lo siguiente:

SCLM_ARCHDEF

Nombre de la archdef referenciada que se está construyendo

AST_SHSCRIPT

Nombre del script RUN de AST para ejecutar AST en z/OS. (Consulte la muestra BWBASTR)

PROJECT NAME

El nombre de proyecto J2EE (no el nombre de proyecto SCLM)

JAR_FILE_NAME

Para el proyecto Java, el nombre del JAR generado

WAR_NAME

Para el proyecto web, el nombre del WAR generado

EJB_NAME

Para el proyecto EJB, el nombre del JAR generado

EAR_NAME

Para la aplicación empresarial, el nombre del EAR generado que se puede desplegar

SCLM_BLDMAP

Si es YES, se incluye en el directorio MANIFEST en JAR, WAR y EAR. Proporciona una correlación de auditoría y construcción de las partes incluidas.

También se incluyen las rutinas ANT de AST para generar el proyecto necesario. Los scripts de construcción SCLM necesarios son versiones modificadas de las muestras AST suministradas a continuación:

- ProjectImport
- ProjectBuild
- Jar
- EJBexport
- WARexport
- EARexport

Rutinas AST no modificadas

Las rutinas que encontrará más abajo se han extraído de la Guía de Application Server Toolkit.

projectImport: Esta tarea importa un proyecto de sistema de archivos existente en un espacio de trabajo.

Tabla 13. Parámetros projectImport

Atributo	Descripción	Necesario
ProjectName	Nombre del proyecto que debe importarse	Sí
ProjectLocation	La ubicación completa del proyecto (o bien en el espacio de trabajo, o bien en cualquier otro sitio del sistema de archivos).	No, el valor predeterminado es \${ubicaciónEspacioTrabajo}/ \${NombreProyecto}

Ejemplo: Importar un proyecto que está en el directorio del espacio de trabajo, pero que actualmente no está en el espacio de trabajo:

```
<projectImport
  ProjectName="myProject"/>
```

projectBuild: Esta tarea construye el proyecto especificado.

Tabla 14. Parámetros projectBuild

Atributo	Descripción	Necesario
ProjectName	Nombre del proyecto que debe construirse	Sí
BuildType	Tipo de construcción	No, el valor predeterminado es Incremental. Puede ser Incremental o Completo.
FailOnError	Si las construcciones deben fallar si se produce un error	No, el valor predeterminado es true (verdadero).

Tabla 14. Parámetros projectBuild (continuación)

Atributo	Descripción	Necesario
DebugCompilation	Si las compilaciones se deberían depurar	No, el valor predeterminado es true (verdadero).
Quiet (en desuso)	Si se deben imprimir mensajes	No, el valor predeterminado es false (falso).
ShowErrors	Si se deben mostrar los errores de proyecto en el registro de construcción Ant	No, el valor predeterminado es true (verdadero).
SeverityLevel	El nivel de problema que debe contarse y tratarse como un error de construcción	No, el valor predeterminado es ERROR. Puede ser ERROR o AVISO o INFORMACIÓN.
CountValidationErrors	Si se deben contar los problemas de validación como errores de proyecto	No, el valor predeterminado es true (verdadero).
PropertyCountName	Propiedad que debe recibir el recuento de errores del proyecto	No, el valor predeterminado es ProjectErrorCount .
PropertyMessagesName	Propiedad que debe recibir los mensajes de error del proyecto	No, el valor predeterminado es ProjectErrorMessages.

Ejemplos:

- Construcción "myProject". El valor predeterminado es incremental con información sobre la depuración:

```
<projectBuild projectName="myProject" />
```

- Efectuar una construcción de producción de "myProject", una construcción completa sin información de depuración:

```
<projectBuild
  projectName="myProject"
  failOnError="true"
  debugCompilation="false"
  buildType="full" />
<echo message="projectBuild: projectName=${projectName}
project Error Count=${ProjectErrorCount}
project Error Messages=${ProjectErrorMessages}" />
```

EJBexport: Esta tarea realiza la misma operación que el asistente de exportación de archivos JAR EJB para exportar un proyecto EJB en un archivo JAR EJB. Esta tarea no está disponible en aquellos productos que no incluyen herramientas de desarrollo EJB.

Tabla 15. Parámetros EJBexport

Atributo	Descripción	Necesario
EJBProjectName	Nombre del proyecto EJB (sensible a mayúsculas y minúsculas)	Sí
EJBExportFile	Vía de acceso absoluta para el archivo JAR EJB.	Sí
ExportSource	Si se deben incluir archivos de origen o no.	No, el valor predeterminado es false (falso).
Overwrite	Si se debe sobrescribir en caso de que el archivo ya exista.	No, el valor predeterminado es false (falso).

Ejemplo: Exportar el proyecto "EJBProject" en "EJBProject.jar"

```
<ejbExport
EJBProjectName="EJBProject"
EJBExportFile="EJBProject.jar"/>
```

WARExport: Esta tarea realiza la misma operación que el asistente de exportación del archivo WAR para exportar un proyecto web en un archivo WAR.

Tabla 16. Parámetros WARExport

Atributo	Descripción	Necesario
WARProjectName	Nombre del proyecto web (sensible a mayúsculas y minúsculas)	Sí
WARExportFile	Vía de acceso absoluta del archivo WAR	Sí
ExportSource	Si se deben incluir archivos de origen o no.	No, el valor predeterminado es false (falso).
Overwrite	Si se debe sobrescribir en caso de que el archivo ya exista.	No, el valor predeterminado es false (falso).

Ejemplo: Exportar el proyecto "ProjectWeb" en "ProjectWeb.war"

```
<warExport WARProjectName="ProjectWeb" WARExportFile="ProjectWeb.war"/>
```

EARExport: Esta tarea realiza la misma operación que el asistente de exportación del archivo WAR para exportar un proyecto web en un archivo WAR.

Tabla 17. Parámetros EARExport

Atributo	Descripción	Necesario
EARProjectName	Nombre del proyecto de aplicación empresarial (sensible a mayúsculas y minúsculas)	Sí
EARExportFile	Vía de acceso absoluta del archivo EAR	Sí
ExportSource	Si se deben incluir archivos de origen o no.	No, el valor predeterminado es false (falso).
IncludeProjectMetaFiles	Si se deben incluir los archivos de metadatos del proyecto que incluyen la vía de acceso de construcción Java, los nombres de proyecto, etc. Se utiliza cuando se reimportan como proyectos binarios.	No, el valor predeterminado es false (falso).
Overwrite	Si se debe sobrescribir en caso de que el archivo ya exista.	No, el valor predeterminado es false (falso).

Ejemplo: Exportar el proyecto "EARProject" en "EARProject.ear"

```
<earExport EARProjectName="EARProject" EARExportFile="EARProject.ear"/>
```

AppClientExport: Esta tarea realiza la misma operación que el asistente de exportación del archivo WAR para exportar un proyecto web en un archivo WAR.

Tabla 18. Parámetros AppClientExport

Atributo	Descripción	Necesario
AppClientProjectName	Nombre del proyecto de cliente de aplicaciones (sensible a mayúsculas y minúsculas)	Sí
AppClientExportFile	Vía de acceso absoluta del archivo JAR de cliente de aplicaciones	Sí

Tabla 18. Parámetros AppClientExport (continuación)

Atributo	Descripción	Necesario
ExportSource	Si se deben incluir archivos de origen o no.	No, el valor predeterminado es false (falso).
Overwrite	Si se debe sobrescribir en caso de que el archivo ya exista.	No, el valor predeterminado es false (falso).

Ejemplo: Exportar el proyecto "ProjectClient" en "ProjectClient.jar":

```
<appClientExport
AppClientProjectName="ProjectClient"
AppClientExportFile="ProjectClient.jar"/>
```

Script de ejecución AST (muestra BWBASTR): Los scripts de construcción AST hacen referencia al script de ejecución AST (type=J2EEBLD lang=TEXT).

```
/*
```

Es un script de construcción de muestra para construir proyectos J2EE utilizando Application Server toolkit (AST) en z/OS.
AST es el programa de creación de Eclipse en modalidad autónoma y es un elemento que se puede instalar por separado fuera de SCLM Developer toolkit.
Este script de muestra deberá personalizarse y debería colocarse en el tipo SCLM del J2EEBLD que se está invocando desde el miembro archdef J2EE mediante la palabra clave archdef SINC. Deberá almacenarse con un lenguaje de texto como TEXT o J2EEPART.

Los argumentos BUILDFILE y WORKSPACE se pasan internamente desde el conversor de lenguaje AST de J2EE en SCLM Developer Toolkit

```
*/
```

```
parse arg $args
parse var $args BUILDFILE WORKSPACE
```

```
/*----- Personalización de usuario -----*/
```

```
/* Personalizar los siguientes valores variables : AST_DIR and JAVA_DIR */
```

```
/* AST_DIR es el directorio de Eclipse z/OS donde se ha instalado AST */
AST_DIR='/u/AST/eclipse'
/* JAVA_DIR es el directorio bin Java z/OS adecuado */
/* Establecer en el bin Java que esté empaquetado en la entrega AST */
JAVA_DIR='/u/AST/eclipse/jdk/bin'
```

```
/*----- Fin de personalización de usuario -----*/
/* El resto de la muestra no debería modificarse */
```

```
say 'AST Eclipse directory = 'AST_DIR
say 'Java bin directory = 'JAVA_DIR
say 'BUILDFILE = 'BUILDFILE
say 'WORKSPACE = 'WORKSPACE
```

```
If SUBSTR(WORKSPACE,1,1) /= '/' Then
Do
say 'ERROR : Invalid WORKSPACE ... Build terminated'
exit 8
Final
```

```
/* Los parámetros siguientes se establecen para la invocación de Eclipse en modalidad autónoma */
/* Opciones de parámetros de memoria Java establecidos en un mín. de 256M & máx. de 512M */
/* Incrementar parámetros de memoria máximos si se produce anomalía de memoria Java */
```

```
M_parm = '-Xms256m -Xmx512m'
```

```

A_parm = '-Dfile.encoding=ISO8859-1 -Xnoargsconversion'
D_parm = '-Dwtp.autotest.noninteractive=true'
cp_parm = '-cp 'AST_DIR'/startup.jar org.Eclipse.core.launcher.Main'
ap_parm = '-application com.ibm.etools.j2ee.ant.RunAnt'
w_parm = '-data "'workspace'"'
f_parm = '-f 'buildfile

PARMS = M_parm' 'A_parm' 'D_parm' 'cp_parm' 'ap_parm' 'w_parm' 'f_parm

/* Las opciones de volcado Java se han desactivado para evitar volcados Java */
/* en anomalías de asignación de memoria */

JAVA_NODUMP='export JAVA_DUMP_OPTS="ONANYSIGNAL(NONE)"'
JAVA_CMD = JAVA_DIR'/java 'PARMS
AST_BUILD = JAVA_NODUMP' ; 'JAVA_CMD

say "Invoking java launch of Eclipse ..."
say AST_BUILD
say "Executing ..."

AST_BUILD
ASTrc = rc
If ASTrc = 23 Then
    say 'WARNING: UNINITIALIZED workspace'
Else
    If ASTrc = 15 Then
        say 'ERROR : WORKSPACE is already BEING USED'

If ASTrc /= 0 then
    Do
        say 'ERROR : BUILD FAILED - return code = 'ASTrc
        EXIT 8
    Final

EXIT 0

```

Script de construcción Java/JAR (muestra): El script de construcción siguiente es type=J2EEBLD lang=J2EEAST. Puede ser cualquier nombre (de hasta 8 caracteres). Variables definidas en formato XML estándar.

<!--

BWBASTJ: Muestra de JAR J2EE para AST

Se trata de un script XML de construcción de muestra que debe ejecutarse utilizando la modalidad autónoma de Eclipse AST en z/OS. El script de construcción SCLM se copiará en el directorio de área de trabajo adecuado del sistema de archivos USS de z/OSS. La palabra clave projectlocation se preformateará dinámicamente con el espacio de trabajo asignado apropiado. Asegúrese de que la línea projectlocation quede tal cual es :

Personalizar el proyecto y las variables de proyecto que siguen

```

nombre proy. : Cambiar ASTJAR por el nombre de proyecto de la aplicación Java
AST_SHSCRIPT : Nombre de script de ejecución AST de esqueleto para la construcción
SCLM_ARCHDEF : Nombre de la archdef que debe construirse
JAR_FILE_NAME : Nombre del JAR creado
SCLM_BLDMAP : Si es YES (sí), debe incluirse en el directorio MANIFEST del JAR.
               Proporciona la correlación de auditoría y construcción de las partes incluidas

```

-->

```

<project name="ASTJAR" default="init" basedir=".">
<property name="AST_SHSCRIPT" value="ASTRUN"/>
<property name="SCLM_ARCHDEF" value="JAR1"/>
<property name="JAR_FILE_NAME" value="ASTjar.jar"/>

```

```

<property name="SCLM_BLDMAP" value="NO"/>
<target name="init">

    <projectImport projectname="${ant.project.name}"
        projectlocation="$WORKSPACE" />
    <Eclipse.refreshLocal resource="${ant.project.name}" depth="infinite" />
    <echo message="refresh done" />

    <projectBuild ProjectName="${ant.project.name}" failonerror="true"
        DebugCompilation="true" BuildType="full" />

    <jar update="true" destfile="${JAR_FILE_NAME}">
    <fileset dir="." excludes="**/*.java"/>
    </jar>

</target>
</project>

```

Script de construcción WAR (muestra): El script de construcción siguiente es
type=J2EEBLD lang=J2EEAST.

```
<!--
```

BWBASTW: Muestra de J2EE para AST

Se trata de un script XML de construcción de muestra que debe ejecutarse
utilizando la modalidad autónoma de Eclipse AST en z/OS.

El script de construcción SCLM se copiará en el directorio de área de trabajo adecuado
del sistema de archivos USS de z/OS.

La palabra clave projectlocation se preformateará dinámicamente con el espacio de trabajo
asignado apropiado. Asegúrese de que la línea projectlocation quede tal cual es :

Personalizar el proyecto y las variables de proyecto que siguen

```

nombre proy. : Cambiar ASTWAR por el nombre de proyecto de la aplicación WAR
AST_SHSCRIPT : Nombre de script de ejecución AST de esqueleto para la construcción
SCLM_ARCHDEF : Nombre de la archdef que debe construirse
WAR_NAME     : Nombre del WAR creado
SCLM_BLDMAP  : Si es YES (sí), debe incluirse en el directorio MANIFEST del WAR.
                Proporciona la correlación de auditoría y construcción de las partes incluidas

```

```
-->
```

```

<project name="ASTWAR" default="init" basedir=".">
<property name="AST_SHSCRIPT" value="BWBASTR"/>
<property name="SCLM_ARCHDEF" value="WAR1"/>
<property name="WAR_NAME" value="ASTwar.war" />
<property name="SCLM_BLDMAP" value="NO"/>
<target name="init">

    <projectImport projectname="${ant.project.name}"
        projectlocation="$WORKSPACE" />
    <Eclipse.refreshLocal resource="${ant.project.name}" depth="infinite" />
    <echo message="refresh done" />

    <projectBuild ProjectName="${ant.project.name}" failonerror="true"
        DebugCompilation="true" BuildType="full" />

    <warExport WARProjectName="${ant.project.name}"
        ExportSource="true"

```

```

        WARExportFile="${WAR_NAME}"/>
    </target>
</project>

```

Script de construcción EJB (muestra): El script de construcción siguiente es type=J2EEBLD lang=J2EEAST.

```
<!--
```

BWBASTEJ: Muestra de EJB J2EE para AST

Se trata de un script XML de construcción de muestra que debe ejecutarse utilizando la modalidad autónoma de Eclipse AST en z/OS. El script de construcción SCLM se copiará en el directorio de área de trabajo adecuado del sistema de archivos USS de z/OSS. La palabra clave projectlocation se preformateará dinámicamente con el espacio de trabajo asignado apropiado. Asegúrese de que la línea projectlocation quede tal cual es :

Personalizar el proyecto y las variables de proyecto que siguen

```

    nombre proy.   : Cambiar ASTEJB por el nombre de proyecto de la aplicación EJB
    AST_SHSCRIPT   : Nombre de script de ejecución AST de esqueleto para la construcción
    SCLM_ARCHDEF   : Nombre de la archdef que debe construirse
    EJB_NAME       : Nombre del JAR EJB creado
    SCLM_BLDMAP    : Si es YES (sí), debe incluirse en el directorio MANIFEST del JAR, WAR o EAR.
                   Proporciona la correlación de auditoría y construcción de las partes incluidas
-->

```

```

<project name="ASTEJB" default="init" basedir=".">
  <property name="AST_SHSCRIPT" value="ASTRUN"/>
  <property name="SCLM_ARCHDEF" value="EJB1"/>
  <property name="EJB_NAME" value="ASTejb.jar" />
  <property name="SCLM_BLDMAP" value="NO"/>
  <target name="init">

    <projectImport projectName="${ant.project.name}"
      projectlocation="$WORKSPACE" />
    <Eclipse.refreshLocal resource="${ant.project.name}" depth="infinite" />
    <echo message="refresh done" />

    <projectBuild projectName="${ant.project.name}" failonerror="true"
      DebugCompilation="true" BuildType="full" />

    <ejbExport ExportSource="true"
      EJBProjectName="${ant.project.name}"
      EJBExportFile="${EJB_NAME}"/>

  </target>
</project>

```

Script de construcción EAR (muestra): El script de construcción siguiente es type=J2EEBLD lang=J2EEAST.

```
<!--
```

BWBASTE: Muestra de EAR J2EE para AST

Se trata de un script XML de construcción de muestra que debe ejecutarse utilizando la modalidad autónoma de Eclipse AST en z/OS. El script de construcción SCLM se copiará en el directorio de área de trabajo adecuado del sistema de archivos USS de z/OSS. La palabra clave projectlocation se preformateará dinámicamente con el espacio de trabajo asignado apropiado durante el tiempo de construcción.

Asegúrese de que la línea projectlocation quede tal cual es.

Personalizar el proyecto y las variables de proyecto que siguen

```
nombre proy. : Cambiar ASTEAR por el nombre de proyecto de la aplicación EAR
AST_SHSCRIPT : Nombre de script de ejecución AST de esqueleto para la construcción
SCLM_ARCHDEF : Nombre de la archdef que debe construirse
EAR_NAME     : Nombre del EAR creado
SCLM_BLDMAP  : Si es YES (sí), debe incluirse en el directorio MANIFEST del JAR, WAR o EAR.
                Proporciona la correlación de auditoría y construcción de las partes incluidas
```

-->

```
<project name="ASTEAR" default="init" basedir=".">
<property name="AST_SHSCRIPT" value="BWBASTR"/>
<property name="SCLM_ARCHDEF" value="EAR1"/>
<property name="EAR_NAME" value="ASTear.ear"/>
<property name="SCLM_BLDMAP" value="NO"/>
<target name="init">

    <projectImport projectname="${ant.project.name}"
        projectlocation="$WORKSPACE" />
    <Eclipse.refreshLocal resource="${ant.project.name}" depth="infinite" />
    <echo message="refresh done" />

    <projectBuild projectName="${ant.project.name}" failonerror="true"
        DebugCompilation="false" BuildType="full" />

    <earExport EARProjectName="${ant.project.name}"
        ExportSource="true"
        EARExportFile="${EAR_NAME}" />

</target>
</project>
```

Formato ARCHDEF J2EE: El formato para la ARCHDEF es el mismo que para el proceso de construcción J2EEANT normal.

SINC La fuente incluye el script de construcción J2EEBLD.

INCLD

SCLM incluye el componente J2EE (por ejemplo, fuente Java).

INCL SCLM incluye otra archdef

OUT1

Indica el tipo de objeto J2EE creado para esta archdef:

- J2EEEAR
- J2EEWAR
- J2EEJAR

LIST Es el listado y la auditoría de componentes de resumen de la ARCHDEF construida. Está dentro de TYPE=J2EELIST, bajo el nombre de miembro de archdef.

LKED

Indica la ARCHDEF LEC y proporciona el lenguaje del conversor de la archdef que debe invocarse (para las ARCHDEF J2EE siempre es J2EEOBJ).

```
SINC SCRIPT1      J2EEBLD
INCLD XX000001    SOURCE
INCL  PAULWAR2    ARCHDEF
```

```
OUT1 * J2EEEAR
LIST *      J2EELIST
LKED  J2EEOBJ
```

Soporte para la construcción SQLJ

SCLM proporciona soporte SQLJ mediante los conversores de construcción Java/J2EE que se suministran con SCLM Developer Toolkit. Estos conversores, en conjunción con los scripts de construcción AST, proporcionan soporte para almacenar y construir proyectos sqlj. También suministra puntos de integración para permitir que las rutinas de cliente establezcan propiedades de personalización SQLJ DB2 y soporten el proceso de enlace DB2 mediante los pasos de construcción y promoción SCLM vía las salidas de construcción y promoción SCLM.

Para obtener información adicional sobre el soporte SQLJ, consulte la Guía del usuario de SCLM Developer Toolkit.

El soporte SQLJ en SCLM (utilizando el método AST) se puede resumir así:

- Almacenamiento de fuente SQLJ en SCLM y asignación de fuente a un tipo de lenguaje de SQLJ
- Construcción de una archdef Java/J2EE, que puede incluir miembros SQLJ como parte de la archdef. La archdef hará referencia a un script de construcción SQLJ en el tipo J2EEBLD, que contendrá propiedades sqlj que deberán ser establecidas y personalizadas por el usuario (consulte los scripts de muestra BWBASTSQ y BWBASTSE).
- Proceso de enlace DB2 posterior opcional (utilizando archivos generados y DBRM almacenados en SCLM). El proceso de enlace posterior debe ser controlado por salidas de usuario de construcción y promoción. Los archivos DBRM a los que se accede directamente dentro de los conjuntos de datos controlados por SCLM.
- Despliegue del JAR/EAR construido mediante el proceso de despliegue DT de SCLM

Personalización de SQLJ (para el administrador de SCLM)

Requisitos del sistema: Debe haber instalado el soporte SQLJ de DB2. El directorio DB2 siguiente (o parecido, en función del directorio de instalación del sitio) debería estar ubicado en el sistema de archivos USS de z/OS: /usr/lpp/db2/db2810/* (es el directorio v8 de db2). Este directorio será necesario para la personalización del script de construcción SQLJ. Para SQLJ, existirán dependencias de vía de acceso de clase en /usr/lpp/db2/db2810/jcc/classes/sqlj.zip. Para db2sqljcustomize, las dependencias de vía de acceso de clase existirán en /usr/lpp/db2/db2810/jcc/classes/db2jcc.jar.

Conversores de lenguaje:

SQLJ: Se suministra un conversor de lenguaje SQLJ nuevo que debería asignarse como el tipo de lenguaje de todo el código fuente SQLJ almacenado en SCLM. El nuevo conversor requiere que se definan Tipos SCLM adicionales. El conversor nuevo es parecido al conversor JAVA pero contiene definiciones IOTYPE adicionales para los tipos SCLM de salida SQLJSER y DBRMLIB. Si el cliente no desea generar archivos DBRM como parte del paso db2sqljcustomize, puede eliminar esta IOTYPE DBRMLIB de la definición de lenguaje SQLJ. (Consulte la definición de conversor SQLJ de muestra BWBTRANS). Dentro del proyecto PROJDEF, genere el conversor SQLJ nuevo y los tipos adicionales de la siguiente manera:

- **SQLJSER:** Es un tipo necesario para que contenga los archivos de perfil serializados (archivos .ser) creados/personalizados en los pasos sqlj y db2sqljcustomize. Es recomendable definir este conjunto de datos de tipo SCLM como recfm=VB , lrecl= 256.
- **DBRMLIB:** Es un tipo necesario para que contenga los archivos DBRM generados creados en el paso db2sqljcustomize. Este tipo sólo resulta necesario para los clientes que utilicen archivos DBRM generados como parte de su proceso de enlace DB2. Es recomendable definir este conjunto de datos de tipo SCLM como recfm=VB , lrecl= 256.

Personalización de usuario de SQLJ

Script de propiedad de construcción SQLJ: Los clientes deben definir propiedades DB2 SQLJ en el script de construcción SQLJ principal (muestra BWBASTSQ) que se ubicará en el tipo J2EEBLD. Estas propiedades se utilizarán en los pasos “sqlj” y “db2sqljcustomize”.

Nota: El BWBASTSQ esqueleto de SQLJ suministrado es una ampliación del BWBASTJ esqueleto de construcción Java normal. Además, BWBASTSE es una ampliación sqlj del BWBASTEJ de muestra EJB. Los clientes pueden mezclar miembros fuente SQLJ y Java en la misma archdef y utilizar BWBASTSQ. La construcción resultante creará un archivo JAR que contenga todos los archivos de clase (class) y .ser generados.

A continuación encontrará una lista de las definiciones de propiedad “sqlj” y “db2sqljcustomize” necesarias.

Valores de propiedad generales

```
<!-- specify the JAVA bin runtime location -->
<property name="JAVA_BIN" value="/usr/lpp/java/J5.0/bin"/>
```

Sqlj

```
<!-- specify the location of the sqlj & db2sqljcustomize exec routine
bwbsqlc.rex (sample BWBSQLC) -->
<!-- By default this sample should be located or copied to the
SCLM DT install directory -->
<property name="sqlj.exec" value="/etc/SCLMDT/bwbsqlc.rex"/>

<!-- especifique argumentos de propiedad globales debajo para proceso sqlj -->
<property name="sqlj.arg" value="-compile=false -status
-linemap=NO -db2optimize"/>
```

db2sqljcustomize

```
<!-- specify the db2jcc.jar location to be included in the
db2sqljcustomize classpath -->
<property name="db2sqljcustomize.cp" value="/usr/lpp
/db2/db2810/jcc/classes/db2jcc.jar" :./SRC:/usr/lpp/db2810/jcc/
classes/db2jcc_license_cisuz.jar"/ >

<!-- specify global property arguments below for db2sqljcustomize -->
<property name="db2sqljcustomize.arg" value="-automaticbind NO -onlinecheck
YES -staticpositioned YES -bindoptions "ISOLATION(CS)" -genDBRM"/>

<!-- Below is the temporary property file name to be passed to a
User property routine for storing additional argument properties
It requires no tailoring -->

<property name="db2sqljcustomize.propfile" value="user.properties"/>

<!-- Below is the name of an optional user program which will be run
immediately before the db2sqljcustomize process.
```

It dynamically updates a property file db2sqljcustomize.propfile to be used as input to the db2sqljcustomize command
 The db2sqljcustomize routine (property: sqlj.exec) will concatenate and use both argument properties set in this build.xml (db2sqljcustomize.arg) and the argument properties read from the user property file.
 The user routine will be passed the following arguments
 basedir : Base directory which is the workspace directory
 propfile : The name of the property file to create (temporary file to be created in workspace)
 sqlj : All the .ser file names to be processed by db2sqljcustomize

Note: The property file being created needs to be basedir+'/'+propfile
 The properties should be set in the file in the following format
 argument=value (eg: singlepkgname=longname:pkgname)
 (one argument declaration per line)

eg:
 singlepkgname=src/TeSQLJ986_SJProfile0.ser:SQLJ986
 singlepkgname=src/TeSQLJ987_SJProfile0.ser:SQLJ987
 pkgversion=1
 url=jdbc:db2://site1.com:80/MVS01
 qualifier=DBT

If no User program is required specify :
 <property name="sqlj.userpgm" value="NONE"/>

-->

<property name="sqlj.userpgm" value="/u/userdir/BWBSQLD"/>

*** end of property settings ***

Las propiedades de argumento se utilizarían para construir la serie de argumento necesaria en la llamada SQLJ o db2sqljcustomize. Por ejemplo:

```
db2sqljcustomize -automaticbind NO -collection ${db2.collid}
-url ${db2.url} -user ${db2.user} -password ???????
-onlinecheck YES -qualifier ${db2.qual} -staticpositioned YES
-pkgversion ${db2.packversion} -bindoptions "ISOLATION(CS)"
-genDBRM -DBRMDir DBRMLIB
-singlepkgname ${db2.pack}
```

Archdef SCLM (muestra ASTSQLJ):

```
*
*
LKED J2EE0BJ          * Conversor de construcción J2EE
*
* Fuente que debe incluirse en la construcción
*
INCLD XX000001 ASTSQLJ * .classpath
INCLD XX000002 ASTSQLJ * .project
INCLD XX000103 ASTSQLJ * .runtime
INCLD BU000082 ASTSQLJ * build.xml
INCLD DE000155 ASTSQLJ * deploy.xml
INCLD SQ000001 ASTSQLJ * SQLJAntScripts/sqlj.customize.xml
INCLD SQ000002 ASTSQLJ * builder/sqlJava.java
INCLD SQ000003 ASTSQLJ * builder/sqlJava.sqlj
INCLD TE000137 ASTSQLJ * Tester.java
INCLD SQ000004 ASTSQLJ * builder/sqlJava_SJProfile0.ser
*
* Script de construcción y salidas generadas
*
SINC ASTSQLJ J2EEBLD  * J2EE JAR Build script
OUT1  *          J2EEJAR
LIST  *          J2EELIST
```

Script de construcción AST de SCLM (muestra ASTSQLJ):

```
<project name="ASTSQLJ" default="compile" basedir=".">
<property name="AST_SHSCRIPT" value="BWBASTR"/>
<property name="SCLM_ARCHDEF" value="ASTSQLJ"/>
<property name="JAR_FILE_NAME" value="ASTSQLJ.jar"/>

<!-- specify the JAVA bin runtime location -->
<property name="JAVA_BIN" value="/u/java/J5.0/bin"/>

<!-- specify the db2jcc.jar location to be included in the
db2sqljcustomize classpath -->
<property name="db2sqljcustomize.cp" value="/usr/lpp/products/db2/db2810/jcc/
classes/db2jcc.jar:./usr/lpp/products/db2/db2810/jcc/classes/
db2jcc_license_cisuz.jar"/>

<!-- specify global property arguments below for sqlj processing -->
<property name="sqlj.arg" value="-compile=false -status -linemap=NO -db2optimize"/>

<!-- specify global property arguments below for db2sqljcustomize -->
<property name="db2sqljcustomize.arg" value="-automaticbind NO -onlinecheck YES
-bindoptions "ISOLATION(CS)" -genDBRM"/>

<!-- specify the location of the db2sqljcustomize exec routine BWBSQLC -->
<property name="db2sqljcustomize.exec" value="/u/SCLMDT/bwbsqlc.rex&cdqg;/>

<!-- Below is the name of the user program to be run as part of the
db2sqljcustomize process . It dynamically updates a property file
to be used as input to the db2sqljcustomize command -->
<property name="sqlj.userpgm" value="NONE"/>

<target description="Pre-Compile SQLJ Files" name="sqlj">
<echo> SQLJ TRANSLATOR </echo>
<apply executable="java" skipemptyfilesets="true" type="file" failonerror="true"
logerror="true">
<arg line="-Dfile.encoding=ISO8859-1 -Xnoargsconversion"/>
<arg line="sqlj.tools.Sqlj"/>
<arg line="${sqlj.arg}"/>
<!-- SER Files -->
<fileset dir=".">
<patternset>
<include name="**/*.sqlj"/>
</patternset>
</fileset>
</apply>
</target>

<target description="Customize SQLJ Files" name="db2sqljcustomize" depends="sqlj">
<!-- Below just echoes properties into a property file db2 props -->
<apply executable="${db2sqljcustomize.exec}" skipemptyfilesets="true"
parallel="true" type="file" failonerror="true" relative="true">
<arg value="${basedir}"/>
<arg value="${db2sqljcustomize.cp}"/>
<arg value="${sqlj.userpgm}"/>
<arg value="${db2sqljcustomize.propfile}"/>
<arg value="db2sqljcustomize ${db2sqljcustomize.arg}"/>
<arg value="sqlj-source"/>
<!-- SER Files -->
<fileset dir=".">
<patternset>
<include name="**/*.ser"/>
</patternset>
</fileset>
</apply>
</target>

<target name="compile" depends="db2sqljcustomize">
```

```

<projectImport projectName=&odq;ASTSQLJ&cdqg;
  projectLocation="$WORKSPACE" />
<eclipse.refreshLocal resource=&odq;ASTSQLJ&cdqg;depth="infinite" />
<echo message="refresh done" />

<projectBuild ProjectName=&odq;ASTSQLJ&cdqg;failOnError="true"
  DebugCompilation="true" BuildType="full" />

<jar update="true" destfile="${JAR_FILE_NAME}">
  <fileset dir="." excludes="**/*.java"/>
</jar>

</target>
</project>

```

Fuente Java en archivos de archivado

De forma predeterminada, la fuente Java se copia en el espacio de trabajo USS como ASCII antes de que tengan lugar las construcciones de compilación. Esto se produce incluso si la fuente Java se almacena en SCLM como EBCDIC.

Si el usuario solicita que la fuente Java se incluya en el archivo de archivado (JAR), a continuación, la fuente estará en ASCII de forma predeterminada. No obstante, se pueden configurar los scripts de formación para que la fuente Java se copie en el espacio de trabajo y se compile en formato EBCDIC, de forma que si se desea una inclusión de fuente Java, ésta tendrá el formato EBCDIC.

Para incluir la fuente Java en formato EBCDIC, los scripts de construcción siguientes deben modificarse de forma correspondiente, tal como se muestra a continuación:

- En el script XML de construcción principal incluya la línea `<property name="ASTJAVA_ENCODING" value="EBCDIC"/>` y elimine la palabra clave de exclusión (excludes) de la rutina de actualización JAR de la siguiente manera:
`jar update="true" destfile="${JAR_FILE_NAME}"> <fileset dir="." excludes="**/*.java"/>`
- En el script de ejecución AST al que la palabra clave AST_SHSCRIPT hace referencia, elimine la palabra clave `-Dfile.encoding=ISO8859-1 A_parm = 'Dfile.encoding=ISO8859-1 -Xnoargsconversion'`

CASO DE EJEMPLO DE UTILIZACIÓN: 'PlantsByWebSphere'

En WebSphere Application Server existen una serie de proyectos de muestra. Éstos se pueden desplegar utilizando EAR preconstruidos o se pueden ensamblar utilizando un proceso de construcción ANT. Aquí encontrará un ejemplo de utilización de SCLM Developer Toolkit para reincorporar este proyecto en SCLM y construirlo/desplegarlo en z/OS mediante AST como el proceso de construcción. PlantsByWebSphere es un proyecto J2EE que implementa una tienda de compras en línea para centros. Está compuesto por los cuatro componentes siguientes:

- Archivador empresarial (PBWProject)
- Bean de Java empresarial (PlantsByWebSphereEJB)
- Archivador web 1 (PlantsByWebSphereWEB)
- Archivador web 2 (PlantsGalleryWEB)

Este documento describe un caso de ejemplo gracias al que esta fuente se coloca en una estructura de proyecto de Eclipse, se incorpora en Developer Toolkit y se construye y se despliega en el host.

1. TAREA ADMINISTRATIVA: Añadir los siguientes tipos a la definición de proyecto:
 - PLANTEAR - Tipo para el EAR
 - PLANTEJB - Tipo para el EJB
 - PLANTWE1 - Tipo para el primer WAR
 - PLANTWE2 - Tipo para el segundo WAR
 2. TAREA ADMINISTRATIVA: Reconstruir el proyecto (someter trabajo).
 3. TAREA ADMINISTRATIVA: Asignar los conjuntos de datos siguientes:
 - <HLQ>.<DEVGROU>.PLANTEJB
 - <HLQ>.<DEVGROU>.PLANTEAR (grande)
 - <HLQ>.<DEVGROU>.PLANTWE1 (grande)
 - <HLQ>.<DEVGROU>.PLANTWE2 (grande)
 4. Asigne código fuente PlantsByWebSphere en una instalación WAS.
(~root/AppServer/samples/src/PlantsByWebSphere)
 5. Inicie el producto Eclipse habilitado para Developer Toolkit para que se pueda elegir.
 6. Importe los 4 componentes en 4 proyectos en el espacio de trabajo de Eclipse.
 7. Asegúrese de que los archivos .project y .settings se incluyan y se establezcan de forma correcta para cada proyecto.
 8. Establezca los valores del compilador y otros archivos de configuración específicos de Websphere. (Proyecto... Propiedades)
 - Establezca facetas Java en 1.4
 - Asegúrese de que se asocie el apéndice de tiempo de ejecución WAS al proyecto
 9. AÑADIR a SCLM (Equipo->Añadir a SCLM en nodo de proyecto)
 - a. Los lenguajes que deben utilizarse son los siguientes:
 - Imágenes – J2EEBIN
 - Archivos Xml – J2EEPART
 - Valores – J2EEPART
 - Código fuente Java – JAVA
 - b. Los tipos que deben utilizarse son los siguientes:
 - Archivos EAR – PLANTEAR
 - Archivos EJB – PLANTEJB
 - Archivos WAR 1 – PLANTWE1
 - Archivos WAR 2 – PLANTWE2
 - c. Archdef (última página de Añadir a asistente de SCLM)
 - Cada componente obtiene una archdef con palabras clave J2EE y scripts de construcción AST.
 - La ARCHDEF de EAR deberían incluir otras Archdefs de componentes.
- ARCHDEF PLANTEAR (muestra):
- ```

*
*
LKED J2EE0BJ * Conversor de construcción J2EE
*
* Fuente que debe incluirse en la construcción
*
INCL PLANTWE1 ARCHDEF
INCL PLANTWE2 ARCHDEF
INCL PLANTEJB ARCHDEF

```



```

*
INCLD XX000002 PLANTEAR * .project
INCLD OR000005 PLANTEAR * .settings/org.Eclipse.wst.common.component
INCLD OR000006 PLANTEAR * .settings/org.Eclipse.wst.common.project.fa
* cet.core.xml
INCLD BU000147 PLANTEAR * EarContent/build.xml
INCLD BU000148 PLANTEAR * EarContent/Database/PLANTSDB/build.xml
INCLD MA000028 PLANTEAR * EarContent/META-INF/MANIFEST.MF
INCLD AP000004 PLANTEAR * EarContent/META-INF/application.xml
INCLD IB000007 PLANTEAR * EarContent/META-INF/ibm-application-bnd.xmi
INCLD IB000008 PLANTEAR * EarContent/META-INF/ibm-application-ext.xmi
INCLD WA000004 PLANTEAR * EarContent/META-INF/was.policy
INCLD PL000017 PLANTEAR * EarContent/Database/PLANTSDB/PLANTSDB.zip
INCLD OR000001 PLANTEAR * .settings/org.Eclipse.core.resources.prefs
INCLD SE000049 PLANTEAR * EarContent/META-INF/ibmconfig/cells/default
* Cell/security.xml
INCLD VA000001 PLANTEAR * EarContent/META-INF/ibmconfig/cells/default
* Cell/applications/defaultApp/deployments/de
* faultApp/variables.xml
*
* Script de construcción y salidas generadas
*
SINC PLANTEAR J2EEBLD * Script de construcción EAR J2EE
OUT1 * J2EEEAR
LIST * J2EELIST

```

- d. Scripts de construcción con los siguientes modelos de salida que deben denominarse:

- PlantsByWebSphere.ear (PLANTEAR)
- PlantsByWebSphereEJB.jar (PLANTEJB)
- PlantsByWebSphere.war (PLANTWE1)
- PlantsGallery.war (PLANTWE2)

PLANTEAR BUILDSCRIPT (muestra):

```

<project name="PBWProject" default="init" basedir=".">
<property name="AST_SHSCRIPT" value="ASTRUN"/>
<property name="SCLM_ARCHDEF" value="PLANTEAR"/>
<property name="EAR_NAME" value="PlantsByWebSphere.ear"/>
<target name="init">
 <projectImport projectName="PBWProject"
 projectlocation="$WORKSPACE" />
 <Eclipse.refreshLocal resource="PBWProject" depth="infinite" />
 <echo message="refresh done" />
 <projectBuild projectName="PBWProject" failonerror="true"
 DebugCompilation="false" BuildType="full" />
 <earExport EARProjectName="PBWProject"
 EARExportFile="${EAR_NAME}"/>
</target>
</project>

```

PLANTWE1 necesita que EJB esté en la vía de acceso de clases durante el tiempo de construcción. Por ello, modifique el script de construcción PLANTWE1' para añadir una propiedad CLASSPATH\_JARS\_FILES, valor "PlantsByWebSphereEJB.jar".

10. Proyecto de construcción (mediante Archdef PLANTEAR)

11. Ejecución de despliegue:

- a. Prepare el script de acción de despliegue (deploy\_ben.jacl). Este despliegue necesita que se cree un origen de datos/conector/proveedor de correo en WAS, por lo que necesita un script de despliegue personalizado.

```

#-----
#
Sample JAACL script for J2EE application deployment

```



```

#
#-----
#
IBM SCLM Developer Toolkit uses the IBM WebSphere Application Server
wsadmin tool to deploy J2EE applications to WAS running on z/OS. The
wsadmin tool requires a JACL script to guide the deployment process.
Hence the JACL script must be installed under UNIX Systems services
(USS) before the deployment process can be invoked.
This sample JACL script should require no customization.

source /u/WebSphere/V6R0/AppServer/samples/bin/AdminUtil.jacl

proc ex1 {args} {

 #-----
 # set arguments
 #-----

 set app [lindex $args 0]
 set appName [lindex $args 1]
 set cellName [lindex $args 2]
 set nodeName [lindex $args 3]
 set serverName [lindex $args 4]

 #-----
 # set up globals
 #-----
 global AdminConfig
 global AdminControl
 global AdminApp

 #-----
 # -- was a earfile name supplied
 #-----
 if {[llength $app] == 0} {
 puts "deploy: Error -- No application specified."
 return
 }

 #-----
 # -- was the appname supplied
 #-----
 if {[llength $appName] == 0} {
 puts "deploy: Error -- Application name not specified."
 return
 }

 #-----
 # Create J2C Resource Adapter
 #-----

 createJ2CResourceAdapter $nodeName $serverName

 #-----
 # Setup security cell
 #-----
 set secAuthAlias "$cellName/samples"
 set secDescript "JAAS Alias for WebSphere Samples"
 set secUserID "samples"
 set secPassword "slamples"
 createJAASAuthenticationAlias $cellName $secAuthAlias $secDescript
 $secUserID $secPassword

```

```

#-----
Create JDBC Provider
#-----

set templName "Cloudscape JDBC Provider (XA)"
All Samples that need JDBC Provider should use/share this one
set provName "Samples Cloudscape JDBC Provider (XA)"
createJDBCProvider $nodeName $serverName $templName $provName

#-----
Create Datasource
#-----

set templName "Cloudscape JDBC Driver XA DataSource"
set dsName "PLANTSDB"
set dsJNDI "jdbc/PlantsByWebSphereDataSource"
set dsDesc "Data source for the Plants by WebSphere entity beans"
set dsAuthMech "BASIC_PASSWORD"
set dbName "\${APP_INSTALL_ROOT}/${CELL}/PlantsByWebSphere.ear/
Database/PLANTSDB"
set secAuthAlias "N_O_N_E"
set connAttrs "upgrade=true"
createDatasource $nodeName $serverName $provName $templName $dsName
$dsJNDI $dsDesc $dsAuthMech $dbName $secAuthAlias $connAttrs

#-----
Create Connection Factory (use builtin_rra)
#-----

set dsName "PLANTSDB"
set cfName "PLANTSDB_CF"
set cfAuthMech "BASIC_PASSWORD"
set secAuthAlias "N_O_N_E"
set cfi "javax.resource.cci.ConnectionFactory"
createConnectionFactory $nodeName $serverName $provName $dsName $cfName
$cfAuthMech $secAuthAlias $cfi

#-----
Create Mail Session
#-----

set provName "Built-in Mail Provider"
set msName "PlantsByWebSphere Mail Session"
set jndiName "mail/PlantsByWebSphere"
set mailTransportHost "yourcompany.ComOrNet"
set mailFrom "userid@yourcompany.ComOrNet"
createMailSession $cellName $nodeName $provName $msName $jndiName
$mailTransportHost $mailFrom

#-----
Setup options for the deployment
Additional options can be added here as required
For Example:
lappend app_options -update
lappend app_options -appname MyAppName
lappend app_options -contextroot MyAppName
lappend app_options -preCompileJSPs
lappend app_options -defaultbinding.force
for a full list of options please use the AdminApp command
wsadmin [return]
$AdminApp options - generic options
or
$AdminApp options MyApp.ear - valid options for your ear file
lappend app_options -node WXP-KEFA25B
#-----
puts "deploy: installing the application"

set app_options [list -server $serverName]
lappend app_options -node $nodeName

```

```

lappend app_options -verbose
lappend app_options -usedefaultbindings
lappend app_options -deployejb
lappend app_options -deployejb.dbtype DERBY_V10
#-----
Install the application onto the server
#-----
$AdminApp install $app $app_options

#-----
Save all the changes
#-----
puts "deploy: saving the configuration"
$AdminConfig save

#-----
Start the installed application
#-----
puts "Starting the application..."
set appManager [$AdminControl queryNames cell=$cellName,
 node=$nodeName,type=ApplicationManager,
 process=$serverName,*]

$AdminControl invoke $appManager startApplication $appName
puts "Started the application successfully..."

puts "deploy: done."
}

#-----
Main
#-----
if { !($argc == 5) } {
 puts "deploy: This script requires 5 parameter: ear file name,
 application name, cell name, node name and server name"
 puts "e.g.: deploy /WebSphere/AppServer/installableApps/
 jmsample.ear myappl myCell myNode myServer"
} else {
 set application [lindex $argv 0]
 set appName [lindex $argv 1]
 set cellName [lindex $argv 2]
 set nodeName [lindex $argv 3]
 set serverName [lindex $argv 4]

 ex1 $application $appName $cellName $nodeName $serverName
}

```

- b. Generar script de propiedades en procesador frontal e iniciar el despliegue. (Desplegar el esqueleto)

12. Verificar mensajes de despliegue. Si presuponemos que todo ha sido satisfactorio, examinar el proyecto en el servidor WAS.



---

## Apéndice E. BUILD FORGE y SCLM

Esta sección abarca los aspectos de implementación y configuración a tener en cuenta al acceder y utilizar SCLM a través de Build Forge. La sección incluye ejemplos de personalización y prueba para promociones y construcciones SCLM.

---

### Visión general

El servidor de consola Build Forge residirá generalmente en una plataforma Windows y deberá configurarse con una llamada de servicio SCLM en el área de proyecto de consola. Cuando se inicia, este proyecto configurado SCLM se conectará con el servidor de agente Build Forge en z/OS que ya se debe haber iniciado. El plug-in RDz Build Forge permite iniciar por socket los proyectos de consola desde el entorno RDz conectándose al servidor de consola. También se puede acceder a la salida de las ejecuciones de proyecto finalizadas desde el plug-in RDz Build Forge dentro de RDz.

---

### Prerrequisitos

- Plug-in Build Forge V7.1 instalado en RDz 7.6
- Consola/servidor Build Forge V7.1
- Agente Build forge para z V7.1 (src-bfagent-7.1.1.0-0022.tar.gz o una versión posterior)

<http://www-01.ibm.com/support/docview.wss?rs=3099&uid=swg24016541>

**Nota:** Es importante utilizar la versión correcta del plug-in Build Forge con la versión correcta del servidor y el agente z/OS. Las versiones compatibles del plug-in y el agente z se empaquetan con el paquete distribuido del servidor Build Forge principal.

Para actualizar la versión correcta del plug-in, actualice de la ubicación del servidor [http://<console\\_host\\_name>/prism/eclipse/updateSite/site.xml](http://<console_host_name>/prism/eclipse/updateSite/site.xml)

---

### Cómo invocar el agente Build Forge en z/OS

Inicie el agente Buildforge en z/OS. El puerto del agente predeterminado es 5555. Por ejemplo:

```
bfagent -s -f /directorio_de_instalación/bfagent-7.1.1.007/src/bfagent.conf
```

Iniciar la consola BF -> Ir a servidores

Seleccione hostname

(hostname:port)

Probar conexión (mediante autenticación de ID de z/OS)

La respuesta del ejemplo es la siguiente:

*Prueba de agente iniciada*

*Host:hostname Puerto:5555*

*Versión de agente: 7.1.1.007*

*Autenticación: userid*

*Plataforma: os/390 18.00 03*

*Prueba funcional: Bien*

*Prueba de agente realizada (Duración 9s)*

Configure un proyecto en la consola BF y ejecútelo.

## Configuración del servidor de consola Build Forge

La configuración mínima siguiente es necesaria para habilitar la funcionalidad SCLM dentro de Build Forge.

1. Servidor (Configure el servidor para que señale al agente Build Forge para System z)
  - a. Configure la autorización del servidor con ID de usuario / contraseña de z/OS. Pulse **Servidor -> Autenticación de servidor**

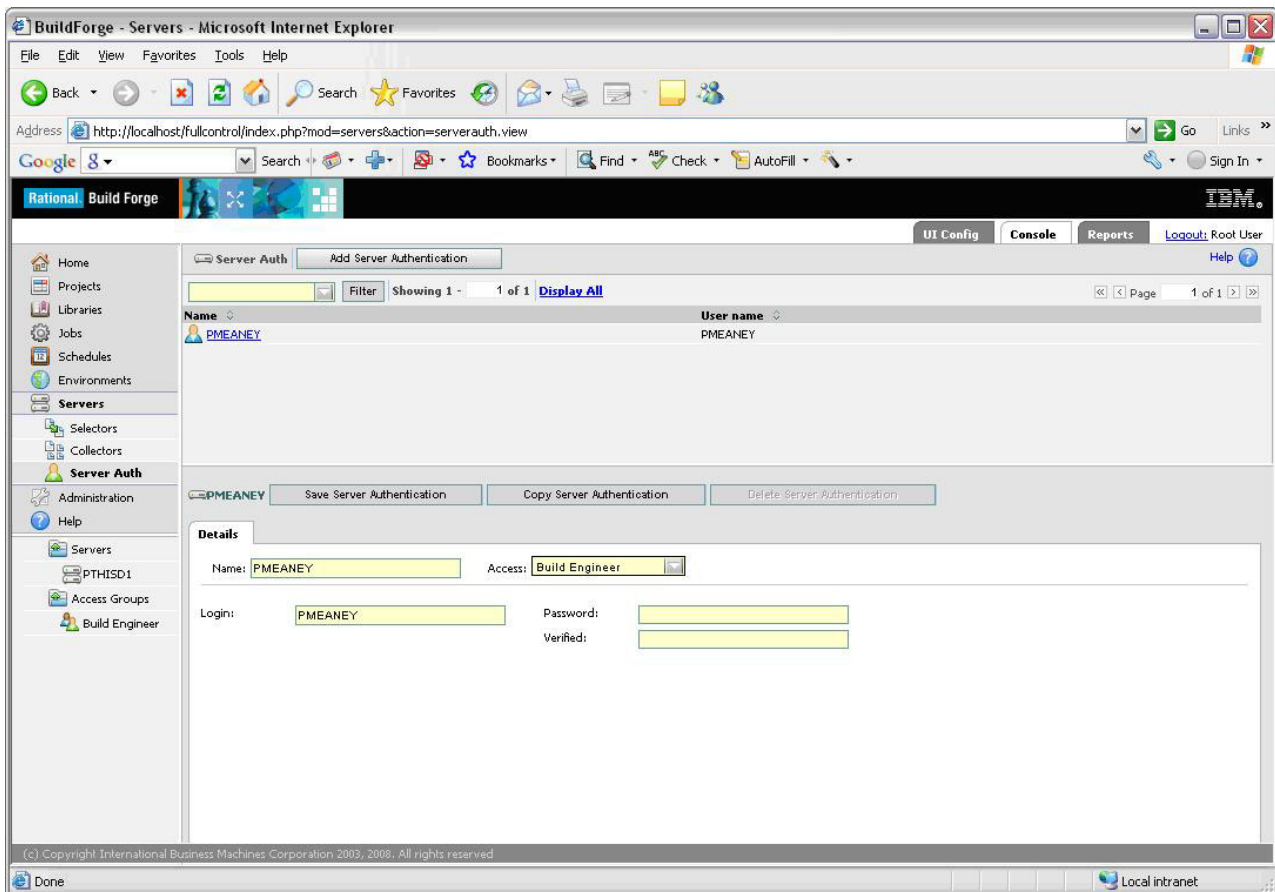


Figura 27. Autorización de servidor

Este ID de usuario debe ser un ID de usuario y una contraseña que exista en el sistema z/OS, en el que se está ejecutando el agente y en el que pretende ejecutar el servicio SCLM. El grupo de acceso será un grupo de acceso existente o un grupo de acceso predeterminado que reside en el servidor de consola con los privilegios adecuados: **Administración -> grupos de acceso**.

- b. Establezca la configuración principal de z/OS Server. (Pulse **Servidores**.)

**NOMBRE:** Nombre exclusivo identificable para su definición de servidor

**PATH:** Este es el directorio de vía de acceso en el host

z/OS en el que se crearán los directorios de construcción. Los directorios de

construcción tendrán normalmente la forma PATH/Nombre de proyecto/BUILD\_#/\*

El archivo de entrada SCLM se creará aquí y la respuesta de salida resultante de la llar aquí según el convenio de denominación especificado en la definición de Proyecto de co

**HOST:** Especifique el nombre de host de destino (DNS o dirección IP)

y el puerto (valor predeterminado del agente z/OS 5555) nombre\_de\_host:puerto  
**ACCESS:** Especifique el tipo de acceso de autorización  
(Por ejemplo: Ingeniero de construcción)

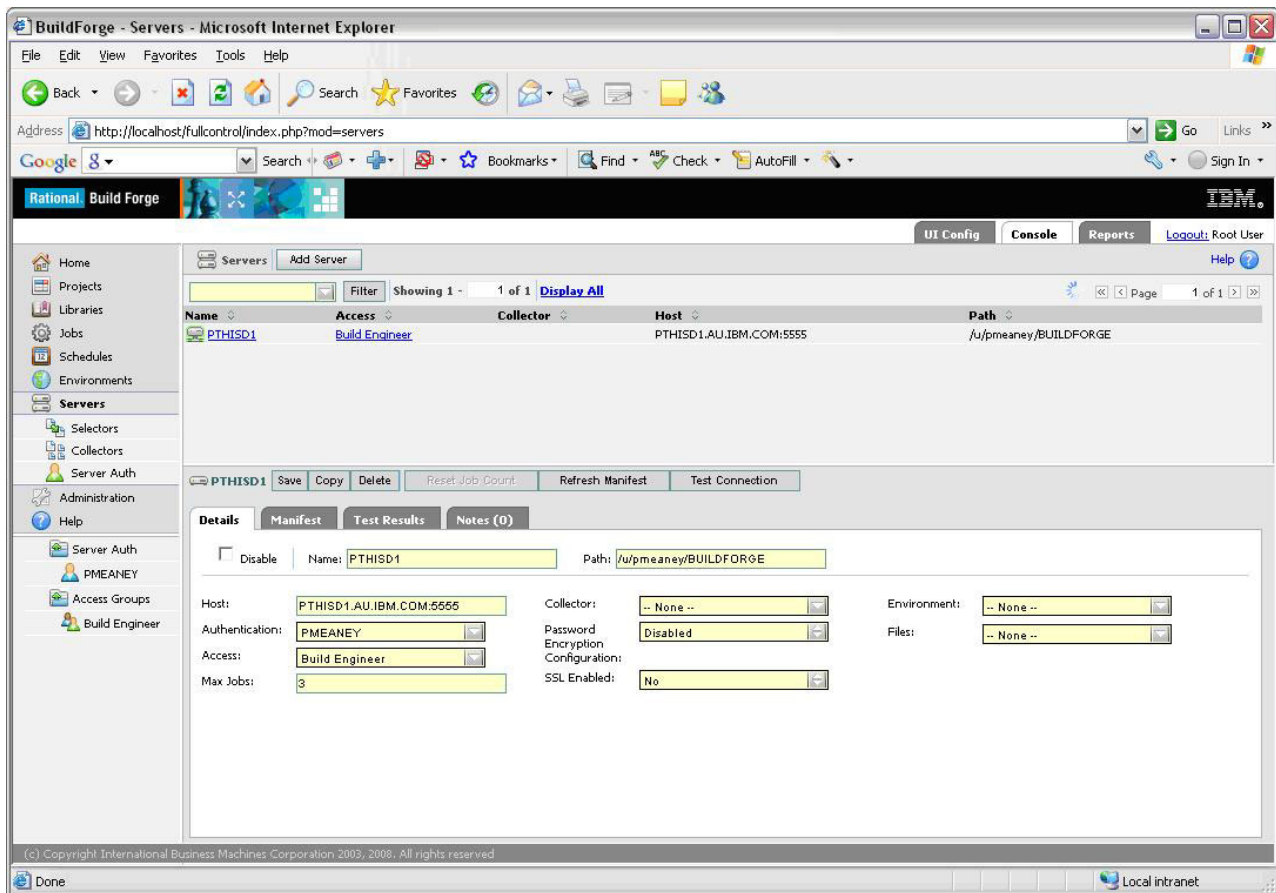


Figura 28. Definición de servidor

Pruebe la conexión con el host pulsando **Probar conexión**. Esto realizará una prueba de conexión con el agente de servidor que se ejecute en z/OS. Espere una prueba satisfactoria de la forma siguiente:

*Prueba de agente iniciada*  
Host: pthisd1.au.ibm.com Port: 5555  
Versión de agente: 7.1.1.007  
Autenticación: IBMUSER  
Plataforma: os/390 18.00 03  
Prueba funcional: Bien  
Prueba de agente realizada (Duración 9s)

- c. Configure el selector principal para el servidor. Pulse **Servidores -> Selector**.

Asigne BF\_NAME al nombre de su servidor. (Por ejemplo: PTHISD1)

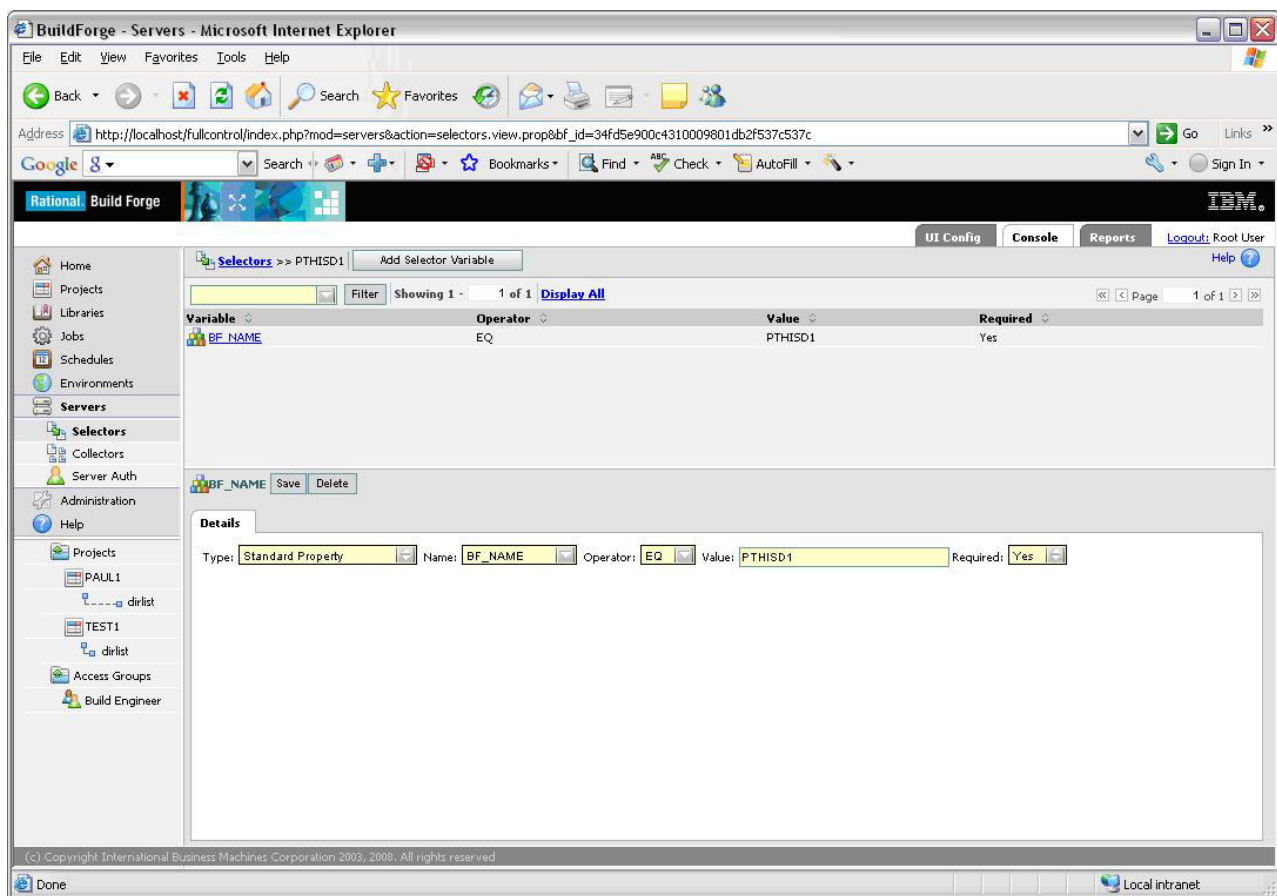


Figura 29. Definición de selector de servidor

2. Configuración de variable de entorno (configure variables de entorno de SCLM Developer Toolkit para el proyecto SCLM.)
  - a. Pulse **Entornos** – Cree un nombre para la lista de variables de entorno SCLM DT. Por ejemplo: SCLMDT  
Configure las variables de entorno SCLM DT siguientes:

**STEPLIB:** El conjunto de

datos STEPLIB en el que residen los módulos de SCLM Developer Toolkit. Esto estará u SFEKLOAD de RDz.

(Por ejemplo: RD4Z.V760.SFEKLOAD) Este tipo de acción en la definición debe ser APPEND.

**\_CMDSERV\_BASE\_HOME:** El directorio base en el que se instaló la pasarela ISPF (p /usr/lpp/ispf) .

**\_CMDSERV\_CONF\_HOME:** El directorio de configuración de la pasarela ISPF (por ej /etc/ispf) .

**\_CMDSERV\_WORK\_HOME:** El directorio de trabajo de la pasarela ISPF (por ejemplo

**\_SCLMDT\_CONF\_HOME:** El directorio de configuración de SCLM DT dentro de RDZ /etc/rd4z760/sclmdt ).

**\_SCLMDT\_WORK\_HOME:** El directorio de trabajo de SCLM DT dentro de RDz. En g mismo valor para esto que para la variable \_CMDSERV\_WORK\_HOME (por ejemplo: \$\_

**PATH:** Las vías de acceso de directorio siguientes -- RDz bin,

\_CMDSERV\_BASE\_HOME bin y el directorio script de sclmdt de RDz.

Esto debe ser el tipo de acción APPEND.



Por ejemplo:

```
/apc/trdz750/usr/lpp/rdz/bin:/etc/rd4z760/sclmdt/CONFIG/script:
$_CMDSERV_BASE_HOME/bin
```

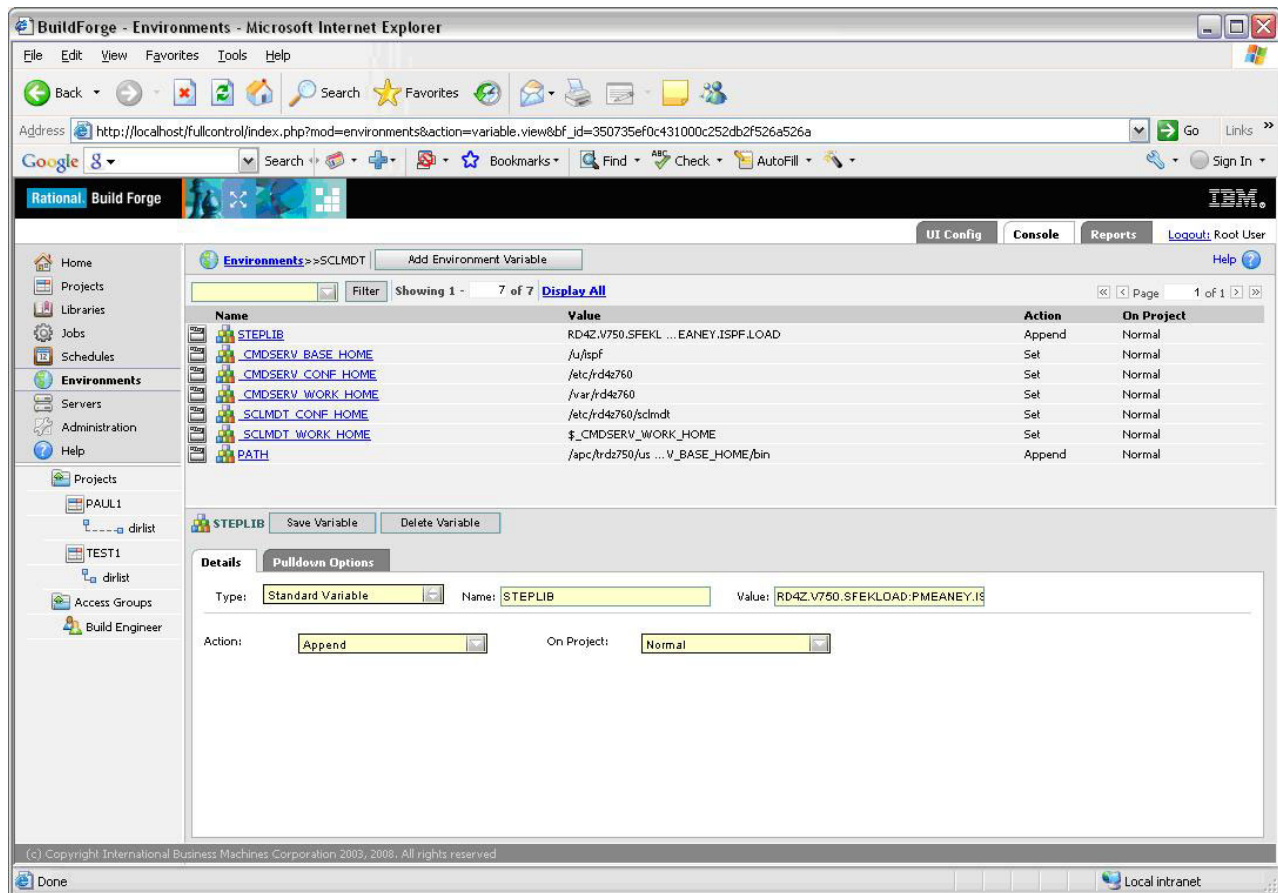


Figura 30. Definiciones de variable de entorno

### 3. Configuración de proyecto SCLM en Build Forge

Ahora debemos configurar los proyectos SCLM para las diferentes peticiones de servicio SCLM. Los ejemplos siguientes serán para la Construcción SCLM y promocionarán servicios que generalmente serían los servicios principales que se ajustarían a la planificación de trabajos de Build Forge. Los ejemplos de script de proyecto están basados en el formato de API XML de SCLM documentado en la *Guía del administrador de SCLM Developer Toolkit*.

- En primer lugar, configure un proyecto SCLM pulsando **Proyectos** -> **Añadir proyecto**.

**NAME:** Especifique el nombre de este proyecto.

(Por ejemplo: SCLM build sample1)

**SELECTOR:** Especifique el nombre del selector configurado en la tabla de selectores (igual que en la sección anterior 1c en la página 125 ).

**ENVIRONMENT:** Especifique el nombre de Entorno configurado para este proyecto SCLM que contiene las variables de entorno SCLM (igual que en la sección anterior 2a en la página 126 ).

Por ejemplo: SCLMDT

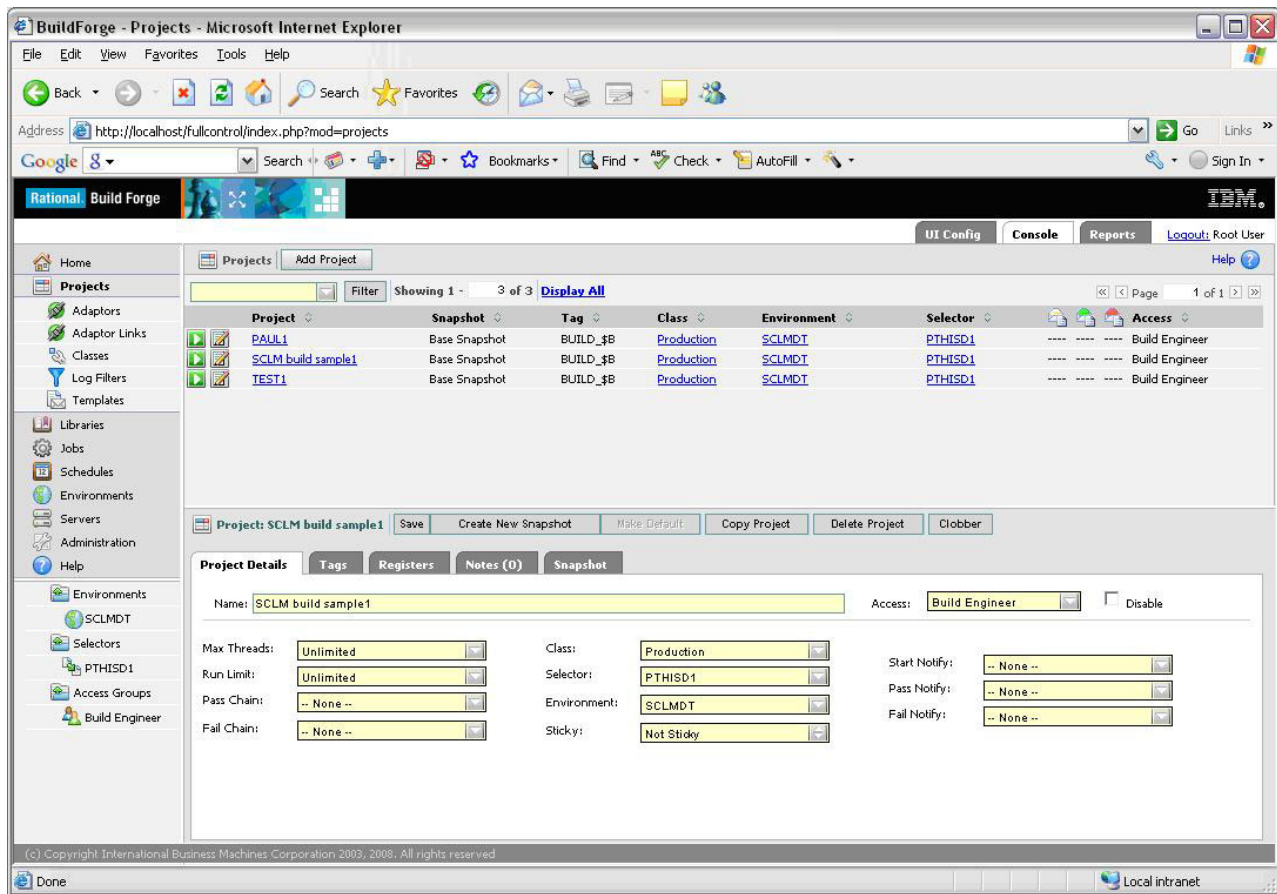


Figura 31. Configuración de ejemplo de proyecto para construcción SCLM (SCLM build sample1)

- b. Ahora añade un paso de construcción a este proyecto SCLM.  
El ejemplo siguiente es un ejemplo de construcción configurado distribuido en el conjunto de datos SAMPLIB de host (BWBBFBLD).

```

echo '<?xml version="1.0"?>' > Build_input.txt
echo '<SCLMDT-INPUT' >> Build_input.txt
echo 'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"' >> Build_input.txt
echo 'xsi:noNamespaceSchemaLocation="sclmdt.xsd">' >> Build_input.txt
echo '<SERVICE_REQUEST>' >> Build_input.txt
echo '<service>SCLM</service>' >> Build_input.txt
echo '<session>NONE</session>' >> Build_input.txt
echo '<ispprof>HLQ.SCLMDT.ISPPROF</ispprof>' >> Build_input.txt
echo '<sclmfunc>BUILD</sclmfunc>' >> Build_input.txt
echo '<project>PROJECT</project>' >> Build_input.txt
echo '<projdef>PROJDEF</projdef>' >> Build_input.txt
echo '<member>MEMBER</member>' >> Build_input.txt
echo '<group>GROUP</group>' >> Build_input.txt
echo '<type>TYPE</type>' >> Build_input.txt
echo '<repdgrp>DEVGRP</repdgrp>' >> Build_input.txt
echo '<groupbld>BLDGRP</groupbld>' >> Build_input.txt
echo '<bldmode>C</bldmode>' >> Build_input.txt
echo '<bldlist>Y</bldlist>' >> Build_input.txt
echo '<bldlstds>NONE</bldlstds>' >> Build_input.txt
echo '<bldrept>Y</bldrept>' >> Build_input.txt
echo '<bldscope>N</bldscope>' >> Build_input.txt
echo '<bldmsg>Y</bldmsg>' >> Build_input.txt
echo '<bldmsgds>NONE</bldmsgds>' >> Build_input.txt
echo '<bldextds>NONE</bldextds>' >> Build_input.txt
echo '</SERVICE-REQUEST>' >> Build_input.txt
echo '<SCLMDT-INPUT>' >> Build_input.txt

cat Build_input.txt | BWBXML >Build_output.txt
cat Build_output.txt

```

Figura 32. \*\* SCLM Build sample1 \*\*

Configure el ejemplo anterior con su proyecto SCLM, grupo, tipo, miembro y sus opciones de construcción tal como se detalla en la sección API de SCLM DT del APÉNDICE C de funcionalidad de Construcción.

Añada este ejemplo configurado al primer paso del proyecto SCLM Build sample1.

**Proyectos -> SCLM Build sample1 -> añadir paso**

**NAME:** El nombre elegido para denominar este paso

**COMMAND:** Incluya el ejemplo configurado más arriba (BWBBFBLD) en este área

El resto de campos pueden permanecer con el valor predeterminado. (Este paso heredaré los valores de entorno del proyecto principal.)

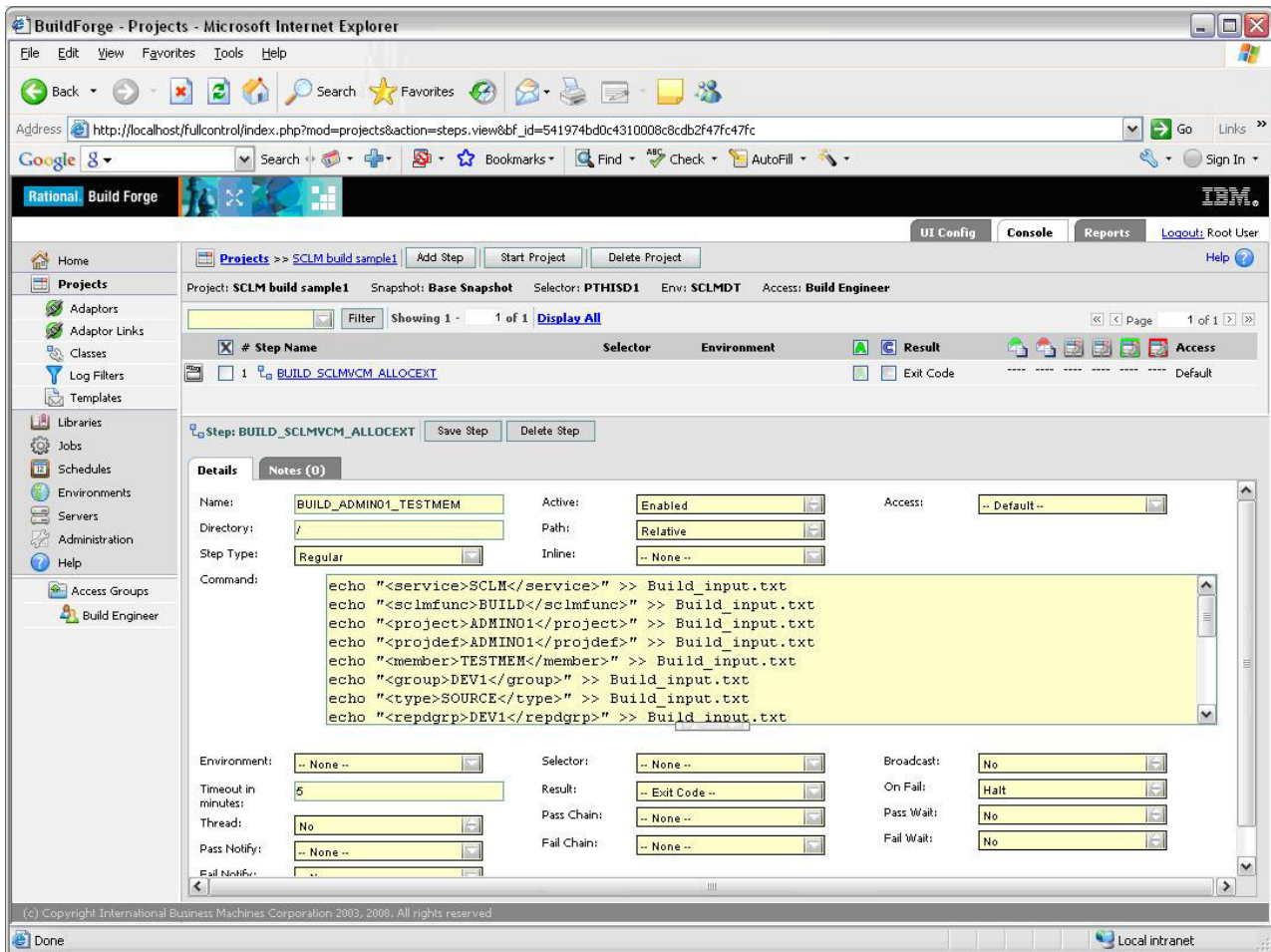


Figura 33. Paso de ejemplo de proyecto en SCLM build sample1

- c. Ejecute el proyecto de construcción SCLM Build sample1.

Pulse **Proyectos -> SCLM Build sample1 -> Iniciar proyecto**. La salida de la construcción se devuelve en la consola **JOBS -> Código de construcción**.

La salida de petición y respuesta también se almacenará en el directorio `z/OS Unix Systems Services` determinado por la variable `PATH` especificada en la configuración del servidor. En el ejemplo anterior, para Server `PTHISD1` con `PATH=/u/IBMUUSER/BUILDFORGE`, los archivos de petición y respuesta se almacenarían bajo el directorio:

```
/u/IBMUUSER/BUILDFORGE/SCLM_Build1_sample/BUILD_1
: >ls
Build_input.txt Build_output.txt
```

Los archivos de petición y respuesta pueden red denominarse dentro del paso de construcción personalizado.

## Ejemplo de promoción SCLM

Los pasos de la sección anterior pueden repetirse para crear un proyecto de promoción SCLM en Build Forge. Sustituya el script de construcción por un ejemplo de script de promoción en el área "COMMAND" del proyecto. El ejemplo siguiente es un ejemplo de promoción configurado distribuido en el conjunto de datos `SAMPLIB` de host (`BWBBFPRM`).

```

echo '<?xml version="1.0"?>' > Promote_input.txt
echo '<SCLMDT-INPUT' >> Promote_input.txt
echo 'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"' >> Promote_input.txt
echo 'xsi:noNamespaceSchemaLocation="sclmdt.xsd">' >> Promote_input.txt
echo '<SERVICE-REQUEST>' >> Promote_input.txt
echo ' <service>SCLM</service>' >> Promote_input.txt
echo ' <session>NONE</session>' >> Promote_input.txt
echo ' <ispprof>HLQ.SCLMDT.ISPPROF</ispprof>' >> Promote_input.txt
echo ' <sclmfunc>PROMOTE</sclmfunc>' >> Promote_input.txt
echo ' <project>PROJECT</project>' >> Promote_input.txt
echo ' <projdef>PROJDEF</projdef>' >> Promote_input.txt
echo ' <member>MEMBER</member>' >> Promote_input.txt
echo ' <group>GROUP</group>' >> Promote_input.txt
echo ' <type>TYPE</type>' >> Promote_input.txt
echo ' <repdgrp>DEVGRP</repdgrp>' >> Promote_input.txt
echo ' <groupprm>PRMGRP</groupprm>' >> Promote_input.txt
echo ' <prmmode>C</prmmode>' >> Promote_input.txt
echo ' <prmrept>Y</prmrept>' >> Promote_input.txt
echo ' <prmscope>N</prmscope>' >> Promote_input.txt
echo ' <prmmmsg>Y</prmmmsg>' >> Promote_input.txt
echo ' <prmmsgds>NONE</prmmsgds>' >> Promote_input.txt
echo ' <prmextds>NONE</prmextds>' >> Promote_input.txt
echo '</SERVICE-REQUEST>' >> Promote_input.txt
echo '</SCLMDT-INPUT>' >> Promote_input.txt

cat Promote_input.txt | BWBXML >Promote_output.txt
cat Promote_output.txt

```

*Figura 34. \*\* SCLM Promote sample1 \*\**

Configure el ejemplo anterior con su proyecto SCLM, grupo, tipo, miembro y las opciones de promoción según se detallan en Apéndice C, “API de SCLM Developer Toolkit”, en la página 65 para la funcionalidad Promoción.



---

## Avisos de la documentación de IBM Rational Developer for System z

© Copyright IBM Corporation - 2009

Derechos restringidos para los usuarios del gobierno de EE. UU. - Utilización, duplicación o divulgación restringidos por el GSA ADP Schedule Contract con IBM Corp.

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
EE.UU.

Para consultas sobre licencias relativas a la información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM en su país o envíe las consultas, por escrito, a:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
3-2-12, Roppongi, Minato-ku, Tokio 106-8711 Japón

El párrafo que sigue no se aplica al Reino Unido ni a ningún otro país en el que tales disposiciones sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, DE COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO DETERMINADO. Algunas legislaciones no contemplan la declaración de limitación de responsabilidad, ni implícitas ni explícitas, en determinadas transacciones, por lo que cabe la posibilidad de que esta declaración no se aplique en su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. La información incluida en este documento está sujeta a cambios periódicos; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento y sin previo aviso.

Cualquier referencia hecha en esta información a sitios web ajenos a IBM se proporciona únicamente para su comodidad y no debe considerarse en modo alguno como promoción de dichos sitios web. Los materiales de dichos sitios web no forman parte de los materiales de este producto de IBM y el usuario será responsable del uso que se haga de dichos sitios web.

IBM puede utilizar o distribuir la información que usted le proporcione del modo que considere conveniente sin incurrir por ello en ninguna obligación para con usted.



Los licenciarios de este programa que deseen obtener información acerca de él con el fin de: (i) intercambiar la información entre los programas creados independientemente y otros programas (incluido este) y (ii) utilizar mutuamente la información que se ha intercambiado, deben ponerse en contacto con:

Intellectual Property Dept. for Rational Software  
IBM Corporation  
3039 Cornwallis Road, PO Box 12195  
Research Triangle Park, NC 27709  
EE.UU.

I

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluyendo en algunos casos el pago de una cantidad.

IBM proporciona el programa bajo licencia descrito en este documento, así como todo el material bajo licencia disponible, según los términos del Acuerdo de Cliente de IBM, del Acuerdo Internacional de Programas bajo Licencia de IBM o de cualquier otro acuerdo equivalente entre ambas partes.

Los datos de rendimiento que se indican en este documento se han obtenido en un entorno controlado. Por lo tanto, los resultados que se obtengan en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas que estén en fase de desarrollo y no existe ninguna garantía de que esas mediciones vayan a ser iguales en los sistemas disponibles en el mercado. Además, es posible que algunas mediciones se hayan estimado por extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información concerniente a productos no IBM se ha obtenido de los suministradores de dichos productos, de sus anuncios publicados o de otras fuentes de información pública disponibles. IBM no ha comprobado dichos productos y no puede afirmar la exactitud en cuanto a rendimiento, compatibilidad u otras características relativas a productos no IBM. Las consultas acerca de las prestaciones de los productos que no son de IBM deben dirigirse a las personas que los suministran.

Todas las declaraciones relacionadas con la dirección o intención futuras de IBM están sujetas a cambio o retirada sin previo aviso, y únicamente representan objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con los nombres y direcciones utilizados por una empresa real es mera coincidencia.

---

## Licencia de copyright

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustra las técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir los programas de ejemplo de cualquier forma, sin tener que pagar a IBM, con intención de desarrollar, utilizar, comercializar o distribuir programas de aplicación que estén en conformidad con la interfaz de programación de aplicaciones (API) de la plataforma operativa para la que están



escritos los programas de ejemplo. Los ejemplos no se han probado minuciosamente bajo todas las condiciones. Por lo tanto IBM, no puede garantizar ni dar por sentada la fiabilidad, la facilidad de mantenimiento ni el funcionamiento de los programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin garantía de ninguna clase. IBM no será responsable de los daños producidos como consecuencia de utilizar los programas de ejemplo.

---

## Reconocimientos de marcas registradas

IBM, el logotipo de IBM e `ibm.com` son marcas registradas de International Business Machines Corp. en muchas jurisdicciones de todo el mundo. Otros productos y nombres de servicio pueden ser marcas registradas de IBM o de otras empresas. Encontrará una lista actual de las marcas registradas IBM en la Web en [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Rational es una marca registrada de International Business Machines Corporation y Rational Software Corporation en los Estados Unidos de América y/o en otros países.

Intel and Pentium son marcas registradas de Intel Corporation en los Estados Unidos de América y/o en otros países.

Microsoft, Windows y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos de América y/o en otros países.

Java y todas las marcas registradas y logotipos basadas en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos de América y/o en otros países.

UNIX es una marca registrada de The Open Group en los Estados Unidos de América y/o en otros países.



# Índice

## Caracteres Especiales

[perfil serializado], establecimiento de enlaces 45  
\$GLOBAL 24

## A

alteraciones temporales, ejemplo de utilización de combinaciones de BIDIPROP 33  
alteraciones temporales, ejemplo de utilización de combinaciones de TRANSLATE.conf 32  
alteraciones temporales de BIDIPROP, ejemplo de utilización de combinaciones de 33  
alteraciones temporales de TRANSLATE.conf, ejemplo de utilización de combinaciones de 32  
API, SCLM Developer Toolkit 65  
API de SCLM Developer Toolkit 65  
ARCHDEF 8, 58  
archivo EAR, DEPLOY 72  
archivo EAR J2EE, DEPLOY 72  
atributos de conjunto de datos, recomendados 7  
atributos para tipos de conjunto de datos habituales, recomendados 7  
autenticación, clave pública 22  
autenticación de clave, pública 22  
autenticación de clave pública 22  
AUTHUPD - cambiar código de autoridad 69

## B

BROWSE - examinar miembro 70  
BUILD - miembro de construcción 70  
BUILD FORGE y SCLM 123  
BWBC9DTJ 17  
BWBCRON1 51  
BWBDEPJ1 17  
BWBDEPJ2 17  
BWBDEPJ3 17  
BWBEARA 17  
BWBEJBA 17  
BWBJAVAA 16  
BWBSQLB 17  
BWBSQLBE 17  
BWBTRANX 17  
BWBWEB 17

## C

CLASSPATH\_JARS 14  
CLASSPATH\_JARS\_FILES 14  
código de autoridad, AUTHUPD 69

combinaciones de alteraciones temporales de BIDIPROP, ejemplo de utilización 33  
combinaciones de alteraciones temporales de TRANSLATE.conf, ejemplo de utilización 32  
Comparación entre JDBC y SQLJ 36  
conceptos, JAVA/J2EE 59  
conceptos, SCLM 57  
conceptos de JAVA/J2EE 59  
conjunto de datos, recuperar LRECL 78  
conjuntos de datos para JAVA/J2EE 7  
conjuntos de datos y miembros, listar los que no son SCLM 78, 79  
conjuntos de datos y miembros que no son de SCLM, listar 78, 79  
construcciones y promociones, iniciadas por CRON 51  
construcciones y promociones iniciadas por CRON 51  
conversión 38  
conversores, lenguaje ASCII/EBCDIC 23  
conversores, tipos SCLM DT y 40  
conversores de lenguaje, ASCII/EBCDIC 23  
conversores de lenguaje, soporte de JAVA/J2EE 5  
conversores de lenguaje ASCII/EBCDIC 23  
conversores de lenguaje EBCDIC 23  
copiar archivo JAR, JARCOPY 77  
correlación de proyectos J2EE con SCLM 18  
correlacionar proyectos J2EE con SCLM, recomendaciones para 18

## D

DB2 35  
DBRM, establecimiento de enlaces 43  
DBRM, qué es 37  
DBRMLIB 9  
definiciones, lenguaje SCLM 5  
definiciones de lenguaje 5  
DELETE - suprimir miembro 71  
denominación, archivos 57  
denominación de archivos 57  
dependencias, CLASSPATH 14  
dependencias de CLASSPATH 14  
DEPLOY - desplegar un archivo EAR J2EE 72  
despliegue, SCLM Developer Toolkit 20  
despliegue, SCLM en UNIX System Services 22  
despliegue, seguro 22  
despliegue, WebSphere Application Server 21  
despliegue de UNIX System Services, SCLM en 22  
despliegue de WebSphere Application Server 21

despliegue seguro 22

## E

EDIT - editar miembro 73  
ejecución, trabajos de construcción de CRON 52  
ejecución de trabajos, construcción de CRON 52  
ejecución de trabajos de construcción, CRON 52  
ejecución de trabajos de construcción de CRON 52  
ejemplos, ARCHDEF J2EE 10  
ejemplos de ARCHDEF J2EE 10  
ejemplos de ARCHDEF samples, J2EE 10  
esqueletos de construcción, XML Ant JAVA/J2EE 16  
esqueletos de construcción XML, Ant JAVA/J2EE 16  
esqueletos de construcción XML Ant, JAVA/J2EE 16  
esqueletos de construcción XML Ant JAVA/J2EE 16  
esquema para mandatos SCLMDT, XML 66  
esquema XML para mandatos SCLMDT 66  
establecimiento de enlaces 39  
establecimiento de enlaces [DBRM] 43  
establecimiento de enlaces [perfil serializado] 45  
estado, recuperar trabajo por lotes 78  
estado de trabajo, recuperar 78  
estado de trabajo por lotes, recuperar 78  
estructura, proyectos SCLM 58  
estructura de proyectos, SCLM 58

## F

FINDLONG, proceso 62  
FINDSHORT, proceso 62  
FORGE y SCLM, BUILD 123  
formato, función 68  
formato, invocación 65  
formato de función 68  
formato de invocación 65  
formatos, miembro SCLM 9  
formatos de miembro, SCLM 9  
formatos de miembro SCLM 9  
funciones y parámetros, de solicitud 68

## G

grupos, recuperar 79

## H

historial, versión 85  
historial de versiones, VERHIST 85

## I

IMPORT, proceso 64  
importar proyecto desde SCLM,  
J2EEIMP 74  
INCL 10  
INCLD 10  
INFO - información sobre el estado del  
miembro 73  
información de proyecto, recuperar 79  
información sobre el estado, INFO 73  
información sobre el estado del miembro,  
INFO 73  
información sobre el miembro,  
UPDATE 83  
informe, DBUTIL 82  
informe DBUTIL, REPUTIL 82  
informe de proyecto, crear 81  
instalación, producto 1  
instalación del producto 1

## J

J2EEANT 5  
J2EEBIN 6  
J2EEBLD 8  
J2EEEAR 9  
J2EEIMP - importar proyecto desde  
SCLM 74  
J2EEJAR 9  
J2EELIST 8  
J2EEMIG - migrar proyecto en SCLM 75  
J2EEMIGB - migrar el proyecto por lotes  
en SCLM 76  
J2EEOBJ 6  
J2EEPART 6  
J2EEWAR 9  
JARCOPY - copiar archivo JAR 77  
JAVA 6  
JAVA/J2EE, conjuntos de datos SCLM  
para 7  
JAVABIN 7  
JAVACLAS 9  
JAVALIST 8  
JDBC 35  
JDBC y SQLJ, comparación 36  
JOBSTAT - recuperar estado de trabajo  
por lotes 78

## L

lenguaje 58  
LIST 10  
lista, funciones 68  
lista de funciones 68  
listar conjuntos de datos y miembros que  
no son de SCLM, MIGDSN 78  
listar conjuntos de datos y miembros que  
no son de SCLM, MIGPDS 79  
LKED 10

LRECL - recuperar LRECL de conjunto de  
datos 78

## M

mandatos, esquema XML para  
SCLMDT 66  
mandatos SCLMDT, esquema XML  
para 66  
miembro, BROWSE 70  
miembro, BUILD 70  
miembro, DELETE 71  
miembro, EDIT 73  
miembro, PROMOTE 80  
miembro, SAVE 82  
miembro, UNLOCK 83  
miembros, listar conjuntos de datos que  
no son de SCLM y 78, 79  
MIGDSN - listar conjuntos de datos y  
miembros que no son de SCLM 78  
MIGPDS - listar conjuntos de datos y  
miembros que no son de SCLM 79  
migrar el proyecto (por lotes),  
J2EEMIGB 76  
migrar el proyecto por lotes,  
J2EEMIGB 76  
migrar proyecto, J2EEMIG 75  
MIGRATE, proceso 64  
Muestras 86  
muestras, sclmdt\_request.xml 86  
muestras, sclmdt\_response.xml 89  
muestras, trabajos de construcción de  
CRON 52  
muestras, xmlbld.java 87  
muestras de trabajos, construcción de  
CRON 52  
muestras de trabajos de construcción,  
CRON 52  
muestras de trabajos de construcción de  
CRON 52

## N

nota sobre la construcción,  
JAVA/J2EE 24  
nota sobre la construcción  
JAVA/J2EE 24

## O

objetos de construcción, JAVA/J2EE 4  
objetos de construcción JAVA/J2EE 4  
opciones, SITE y específicas de  
proyecto 27  
opciones de almacenamiento, ASCII o  
EBCDIC 23  
opciones de almacenamiento ASCII o  
EBCDIC 23  
opciones de almacenamiento  
EBCDIC 23  
opciones de despliegue, otras 23  
opciones específicas de proyecto 27  
otras opciones, despliegue 23  
OUT1 10

## P

parámetros de solicitud, funciones y 68  
perfil serializado 37  
personalización 39  
personalización para SCLM Developer  
Toolkit 3  
personalizado, script de usuario 42  
preparación, programa SQLJ 38  
preparación del programa, SQLJ 38  
preparación del programa SQLJ 38  
proceso, FINDLONG 62  
proceso, FINDSHORT 62  
proceso, IMPORT 64  
proceso, MIGRATE 64  
proceso, TRANSLATE 63  
proceso de construcción 40  
proceso de registros, nombres  
largos/cortos múltiple 63  
proceso de registros, nombres  
largos/cortos único 62  
proceso de registros de nombres,  
largos/cortos múltiple 63  
proceso de registros de nombres,  
largos/cortos único 62  
proceso de registros de nombres  
largos/cortos, múltiple 63  
proceso de registros de nombres  
largos/cortos, único 62  
programa de conversión, resumen técnico  
de SCLM 61  
Programa de utilidad de construcción,  
Rational Application Developer for  
WebSphere Software 97  
Programa de utilidad de construcción de  
Rational Application Developer for  
WebSphere Software. 97  
PROJGRPS - recuperar grupos para un  
proyecto 79  
PROJINFO - recuperar información de  
proyecto 79  
promociones, construcciones iniciadas por  
CRON y 51  
PROMOTE - promover un miembro 80  
propiedades, db2sqljcustomize.\* 41  
propiedades, SCLM 58  
propiedades, sqlj.\* 41  
propiedades de db2sqljcustomize.\* 41  
propiedades de sqlj.\* 41  
proyectos J2EE, correlación con  
SCLM 18  
proyectos J2EE, recomendaciones para  
correlacionar 18

## R

recuperar estado de trabajo por lotes,  
JOBSTAT 78  
recuperar grupos para un proyecto,  
PROJGRPS 79  
recuperar información de proyecto,  
PROJINFO 79  
recuperar LRECL de conjunto de datos,  
LRECL 78  
REPORT - crear informe de proyecto 81  
REPUTIL - informe DBUTIL 82  
requisitos, STEPLIB y PATH 51

- requisitos de PATH, STEPLIB y 51
- requisitos de STEPLIB y PATH 51
- resumen de construcciones,
  - JAVA/J2EE 3
- resumen de construcciones JAVA/J2EE 3
- resumen del programa de conversión de
  - SCLM, técnico 61
- resumen técnico del programa de
  - conversión de SCLM 61

## S

- SAVE - guardar miembro 82
- SCLM, BUILD FORGE y 123
- sclmdt\_request.xml 86
- sclmdt\_response.xml 89
- script, de usuario personalizado 42
- script de construcción, ajuste 41
- scripts, ejemplo J2EE 15
- scripts de ejemplo J2EE 15
- seguridad, SCLM 47
- SINC 10
- SITE y opciones específicas de
  - proyecto 27
- solicitud, funciones y parámetros de 68
- soporte, SQLJ 35
- soporte de JAVA/J2EE, conversores de
  - lenguaje para 5
- soporte SQLJ 35
- SQL 35
- SQLJ 7
- SQLJ, qué es 36
- SQLJ y JDBC, comparación 36
- SQLJSER 9

## T

- tabla, conversión de nombres
  - largos/cortos 61
- tabla de conversión, nombres
  - largos/cortos 61
- tabla de conversión de nombres,
  - largos/cortos 61
- tabla de conversión de nombres
  - largos/cortos 61
- Tipo 57
- tipos, Java/J2EE 9
- tipos, SCLM 7
- tipos Java/J2EE 9
- tipos y conversores, SCLM DT 40
- tipos y conversores de SCLM DT 40
- TRANSLATE, proceso 63

## U

- UNLOCK - desbloquear miembro 83
- UPDATE - actualizar información sobre el
  - miembro 83
- usuario, script personalizado de 42
- utilización de combinaciones de
  - alteraciones temporales de BIDIPROP,
    - ejemplo 33
- utilización de combinaciones de
  - alteraciones temporales de
    - TRANSLATE.conf, ejemplo 32

## V

- variables, definidas por el usuario 13
- variables definidas por el usuario 13
- VERBROW - versiones de
  - navegación 84
- VERDEL - versiones de supresión 84
- VERHIST - historial de versiones de
  - SCLM 85
- VERLIST - versiones de lista 85
- VERREC - versiones de recuperación 86
- versiones, lista 85
- versiones, navegación 84
- versiones, recuperación 86
- versiones, supresión 84
- versiones de lista, VERLIST 85
- versiones de navegación, VERBROW 84
- versiones de recuperación, VERREC 86
- versiones de supresión, VERDEL 84
- visión general, SCLM 57

## X

- xmlbld.java 87



---

# Hoja de Comentarios

IBM Rational Developer for System z  
Guía del administrador de SCLM Developer Toolkit  
Versión 7.6

Número de Publicación SC11-3815-01

Por favor, sírvase facilitarnos su opinión sobre esta publicación, tanto a nivel general (organización, contenido, utilidad, facilidad de lectura,...) como a nivel específico (errores u omisiones concretos). Tenga en cuenta que los comentarios que nos envíe deben estar relacionados exclusivamente con la información contenida en este manual y a la forma de presentación de ésta.

Para realizar consultas técnicas o solicitar información acerca de productos y precios, por favor diríjase a su sucursal de IBM, business partner de IBM o concesionario autorizado.

Para preguntas de tipo general, llame a "IBM Responde" (número de teléfono 901 300 000).

Al enviar comentarios a IBM, se garantiza a IBM el derecho no exclusivo de utilizar o distribuir dichos comentarios en la forma que considere apropiada sin incurrir por ello en ninguna obligación con el remitente.

Comentarios:

Gracias por su colaboración.

Para enviar sus comentarios:

- Envíelos por correo a la dirección indicada en el reverso.
- Envíelos por fax al número siguiente: 1-800-227-5088 (EE.UU. y Canadá)
- Envíelos por correo electrónico a: [kfrye@us.ibm.com](mailto:kfrye@us.ibm.com)

Si desea obtener respuesta de IBM, rellene la información siguiente:

Nombre

Dirección

Compañía

Número de teléfono

Dirección de e-mail

IBM Corporation  
Information Development  
Department G71A / Bldg. 503  
P.O. Box 12195  
Research Triangle Park, NC







Número de Programa: 5724-T07

Impreso en España

SC11-3815-01

