



Version 7.6



SCLM Developer Toolkit - Guide d'administration

Important

Avant d'utiliser cette documentation, reportez-vous aux informations générales sous «Remarques sur la documentation pour IBM Rational Developer for System z», à la page 133.

Remarque

Certaines illustrations de ce manuel ne sont pas disponibles en français à la date d'édition.

Deuxième édition - octobre 2009

Réf. US : SC23-9801-01

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- <http://www.fr.ibm.com> (serveur IBM en France)
- <http://www.can.ibm.com> (serveur IBM au Canada)
- <http://www.ibm.com> (serveur IBM aux Etats-Unis)

*Compagnie IBM France
Direction Qualité
Tour Descartes
92066 Paris-La Défense Cedex 50*

© Copyright IBM France 2009. Tous droits réservés.

© Copyright International Business Machines Corporation 2009.

Table des matières

Tableaux	v
-----------------	----------

A propos de ce document	vii
A qui s'adresse ce guide	vii

Chapitre 1. Installation du produit	1
--	----------

Chapitre 2. Personnalisation de SCLM pour SCLM Developer Toolkit	3
Récapitulatif de la génération JAVA/J2EE	3
Objets de génération JAVA/J2EE générés	4
Traducteurs de langage pour le support JAVA/J2EE	5
Définitions de langage SCLM	5
Fichiers SCLM pour JAVA/J2EE	7
Types SCLM	7
Attributs de fichier recommandés pour certains types standard	7
Formats des membres SCLM	9
Format \$GLOBAL	9
Format d'ARCHDEF J2EE	10
Format du script de génération Ant J2EE	13
Squelettes de génération XML Ant JAVA/J2EE	16
Mappage de projets J2EE à SCLM	18
Recommandations de mappage des projets J2EE à SCLM	19
Déploiement de SCLM Developer Toolkit	20
Déploiement de WebSphere Application Server (WAS)	21
Déploiement de SCLM sur UNIX System Services	22
Déploiement sécurisé	22
Authentification de clé publique	22
Autres options de déploiement	23
Options de stockage ASCII ou EBCDIC	23
Traducteurs de langage ASCII/EBCDIC	23
\$GLOBAL	24
TRANSLATE.conf	26
Options spécifiques au projet et SITE	28
Exemple d'utilisation de combinaisons de remplacements de fichier TRANSLATE.conf	32
Exemple d'utilisation de combinaisons de remplacements de BIDIPROP	33

Chapitre 3. Support SQLJ	35
Présentation de SQL	35
Présentation de DB2	35
Qu'est-ce que JDBC ?	35
Qu'est-ce que SQLJ ?	36
Comparaison de JDBC et SQLJ	36
Présentation d'un profil sérialisé	37
Présentation d'un module DBRM	37
Préparation du programme SQLJ	38
Traduction	38
Personnalisation	39
Liaison	39

Traducteurs et types SCLM DT	40
Personnalisation de la procédure de génération	40
Personnalisation du script de génération	41

Chapitre 4. Sécurité SCLM	47
Option de sécurité	47
Configuration dans votre produit de sécurité	47
Profils de sécurité	47
ID utilisateur de substitution	48
Exemple : Génération	48
Exemple : Déploiement	49

Chapitre 5. Générations et promotions lancées via le service CRON	51
Spécifications de STEPLIB et PATH	52
Exécution des travaux de génération CRON	52
Exemples de travaux de génération CRON	53

Annexe A. Présentation de SCLM	57
Concepts de SCLM	57
Dénomination des fichiers	57
Type	57
Langage	58
Propriétés de SCLM	58
Structure d'un projet SCLM	58
ARCHDEF	58
Concepts de JAVA/J2EE	59

Annexe B. Table de traduction des noms longs/abrévés	61
Récapitulatif technique du programme de traduction SCLM	61
Traitement unique des enregistrements de nom long/abrégé	62
Traitement FINDLONG	62
Traitement FINDSHORT	62
Traitement TRANSLATE	63
Traitement multiple des enregistrements de nom long/abrégé	63
Traitement IMPORT	64
Traitement MIGRATE	64

Annexe C. API de SCLM Developer Toolkit	65
Format d'appel	65
Schéma XML des commandes SCLMDT	66
Paramètres et fonctions des demandes	68
Format des fonctions	68
Liste des fonctions	68
AUTHUPD – Modification du code des droits d'accès SCLM	69
BROWSE – Exploration d'un membre SCLM	70
BUILD – Création d'un membre SCLM	70
DELETE – Suppression d'un membre SCLM	72

DEPLOY – Déploiement d’un fichier EAR J2EE	72
EDIT – Edition d’un membre SCLM	73
INFO – Informations sur le statut du membre SCLM	74
J2EIMP – Importation d’un projet à partir de SCLM	74
J2EEMIG – Migration d’un projet dans SCLM	75
J2EEMIGB – Migration par lots d’un projet dans SCLM	77
JARCOPY – Copie d’un fichier JAR	77
JOBSTAT – Extraction du statut d’un travail par lots	78
LRECL – Extraction du fichier SCLM ou LRECL	78
MIGDSN – Liste des membres et des fichiers non SCLM	79
MIGPDS – Migration des membres et des fichiers NON-SCLM dans SCLM	79
PROJGRPS – Extraction des groupes SCLM d’un projet	80
PROJINFO – Extraction des informations sur un projet SCLM	80
PROMOTE – Promotion d’un membre SCLM	80
REPORT – Création d’un rapport de projet	82
REPUTIL –Rapport SCLM DBUTIL	82
SAVE – Sauvegarde d’un membre SCLM	83
UNLOCK – Déverrouillage d’un membre SCLM	84
UPDATE – Mise à jour des informations d’un membre SCLM	84
VERBROW – Versions d’exploration de SCLM	85
VERDEL – Suppression de versions SCLM	85
VERHIST – Historique des versions SCLM	86
VERLIST – Versions des listes SCLM	86
VERREC – Versions de restauration SCLM	87
Exemples	88
sclmdt_request.xml	88
xmlbld.java	88
sclmdt_response.xml	90

Annexe D. Utilitaire de génération Rational Application Developer for WebSphere Software. 97

Présentation de l’utilitaire de génération Rational Application Developer for WebSphere	97
Stockage d’objets Java/J2EE dans SCLM.	98
Utilitaire de génération comparé à la procédure de génération ANT native	98
Remarques sur l’installation de l’utilitaire de génération Rational Application Developer for WebSphere Software	99
Intégration de SCLM à l’utilitaire de génération	99
Implémentation et exécution de l’utilitaire de génération Rational Application Developer for WebSphere Software	99
Traducteurs de langage AST de SCLM	100
Formats et scripts de génération de l’utilitaire de génération	103
Prise en charge de la génération SQLJ	112
Source Java dans les fichier d’archive	116
SCENARIO D’UTILISATION :	
‘PlantsByWebSphere’	117

Annexe E. BUILD FORGE et SCLM 123

Présentation.	123
Conditions préalables	123
Comment appeler l’agent Build Forge sous z/OS	123
Configuration du serveur de console Build Forge	124
Exemple de promotion SCLM	132

Remarques sur la documentation pour IBM Rational Developer for System z	133
Licence de copyright	135
Marques	135

Index 137

Tableaux

1.	Liste de contrôle de l'administrateur SCLM	1
2.	Traducteurs SCLM Developer Toolkit	5
3.	Variables définies par le client	13
4.	Traducteurs de langage SCLM et ASCII/EBCDIC	23
5.	Variables \$GLOBAL.	25
6.	Options SITE/PROJET	30
7.	Comparaison de JDBC et SQLJ	36
8.	Types de traducteur SCLM pour SQLJ	40
9.	sqlj.* properties	41
10.	db2sqljcustomize.* properties	42
11.	Profils de sécurité SCLMDT Developer Toolkit	48
12.	Paramètres de traduction des noms longs/abrégés.	62
13.	Paramètres projectImport	104
14.	Paramètres projectBuild	104
15.	Paramètres EJBexport	105
16.	Paramètres WARExport	105
17.	Paramètres EARExport	106
18.	Paramètres AppClientExport	106

A propos de ce document

Le présent document aborde la configuration d'IBM® Software Configuration and Library Manager (SCLM) requise pour la fonction SCLM Developer Toolkit d'IBM Rational Developer for System z.

A partir de maintenant, les noms suivants sont utilisés dans le présent ouvrage :

- *IBM Software Configuration and Library Manager* est appelé *SCLM*.
- *IBM Rational Developer for System z* est appelé *Developer for System z*.
- La fonction *SCLM Developer Toolkit* d'*IBM Rational Developer for System z* est appelée *SCLM Developer Toolkit*, et parfois *Developer Toolkit* ou *SCLMDT*.

A qui s'adresse ce guide

Le présent document contient des informations destinées à l'administrateur de projets SCLM qui seront utilisées avec SCLM Developer Toolkit. Cela comprend les projets qui utilisent les langages des composants Java™ et z/OS UNIX® System Services, ainsi que les projets SCLM traditionnels.

Ces administrateurs doivent connaître z/OS UNIX System Services, le script REXX, le compilateur Java et les définitions de langage et de projet SCLM.

Chapitre 1. Installation du produit

Le présent document ne couvre pas l'implémentation et le chargement du produit SCLM, qui est livré avec le système d'exploitation z/OS. Il n'aborde pas non plus l'installation et la configuration de SCLM Developer Toolkit, qui est une fonction de Rational Developer for System z.

Pour plus d'informations sur ces tâches, voir *ISPF Software Configuration and Library Manager Project Manager's and Developer's Guide* (SC34-4817) et *Rational Developer for System z - Guide de configuration de l'hôte* (SC11-6285).

Pour effectuer les tâches de personnalisation et de définition de projet, l'administrateur SCLM doit connaître un certain nombre de valeurs Developer for System z personnalisables, comme indiqué dans le tableau 1. Pour plus d'informations, adressez-vous au programmeur système z/OS chargé d'installer et de personnaliser Developer for System z.

Tableau 1. Liste de contrôle de l'administrateur SCLM

Description	Valeur par défaut	Emplacement de la réponse	Valeur
Exemple de bibliothèque Developer for System z	FEK.SFEKSAMP	Installation SMP/E	
Exemple de répertoire Developer for System z	/usr/lpp/rdz/samples	Installation SMP/E	
Répertoire bin Java	/usr/lpp/java/J5.0/bin	rsed.envvars - \$JAVA_HOME/bin	
Répertoire bin Ant	/usr/lpp/Ant/apache-ant-1.7.1/bin	rsed.envvars - \$ANT_HOME/bin	
Répertoire de base WORKAREA	/var/rdz	rsed.envvars - \$_CMDSERV_CONF_HOME	
Répertoire de base de la configuration des projets SCLMDT	/etc/rdz/sclmdt	rsed.envvars	
VSAM de traduction des noms longs/abrégés	FEK.#CUST.LSTRANS.FILE	rsed.envvars - \$_SCLMDT_TRANTABLE	

Chapitre 2. Personnalisation de SCLM pour SCLM Developer Toolkit

Ce chapitre explique comment l'administrateur SCLM peut personnaliser SCLM pour une utilisation par SCLM Developer Toolkit.

Récapitulatif de la génération JAVA/J2EE

Voici un résumé de la procédure effectuée pour les générations Java et J2EE à l'aide des traducteurs fournis.

Remarque : Vous pouvez générer des membres JAVA/J2EE ou des définitions d'archive (ARCHDEFS) directement dans TSO/ISPF sur l'hôte, mais également via Developer Toolkit Client.

L'ARCHDEF contient les membres qui constituent le projet JAVA/J2EE et qui correspondent à une représentation abrégée de la manière dont le projet existe dans un espace de travail Eclipse.

L'ARCHDEF elle-même est générée, ce qui appelle un traducteur de langage de vérification de pré-génération (J2EEANT). Le traducteur lit le script de génération J2EE, qui est représenté dans l'ARCHDEF par le mot clé SINC et remplace les propriétés spécifiées dans le squelette XML Ant référencé par les propriétés XML SCLM-ANTXML (A). Le script de génération, lorsqu'il est généré par SCLM Developer Toolkit, est stocké dans SCLM avec le langage J2EEANT (1).

Une ARCHDEF génère les classes Java de la source Java identifiée avec le mot clé INCLD dans l'ARCHDEF (2) et chaque ARCHDEF peut également générer un fichier d'archive J2EE, tel qu'un fichier JAR, WAR ou EAR. L'objet J2EE créé dépend du script de génération correspondant référencé et de l'utilisation du mot clé d'ARCHDEF OUT1 (3), (B).

Lorsque l'ARCHDEF est générée, le traducteur de langage de vérification de pré-génération associé au script de génération (dans le type SCLM J2EEBLD) s'exécute. Il détermine quels composants de l'ARCHDEF doivent être à nouveau générés (y compris les ARCHDEF imbriquées (4) identifiées via l'utilisation du mot clé INCL dans l'ARCHDEF). Ces composants sont ensuite copiés dans le répertoire du système de fichiers z/OS UNIX System Services et le script Ant compile, puis génère les objets JAVA/J2EE indiqués par le script de génération et l'ARCHDEF. Les références à des fichiers jar ou des classes externes que votre projet IDE doit résoudre sont résolus à partir du chemin d'accès défini dans la propriété CLASSPATH_JARS (C).

SCLM traite ensuite chaque composant ARCHDEF en exécutant chaque traducteur de langage associé au composant. Le traducteur de langage Java, associé au code source Java, copie les fichiers de classe créés dans SCLM.

Enfin, le traducteur d'ARCHDEF détermine quels objets J2EE ont été générés (JAR, WAR, EAR) et les recopie dans SCLM.

Il est essentiel de créer une ARCHDEF séparée pour chaque composant d'application qui peut constituer une application d'entreprise (EAR). En d'autres

termes, un fichier EAR contenant un fichier WAR, qui contient à son tour un EJB JAR, doit disposer d'une ARCHDEF JAR, d'une archdef WAR avec une inclusion de l'ARCHDEF EJB JAR. L'ARCHDEF EAR doit ensuite inclure une instruction INCL de l'ARCHDEF WAR.

L'exemple suivant montre le code JAR :

```
*
* Initially generated on 10/05/2006 by SCLM DT V2
*
LKED J2EE0BJ          * J2EE Build translator
*
* Source to include in build
*
INCLD AN000002 V2TEST * com/Angelina.java          *
INCLD V2000002 V2TEST * com/V2Java1.java (2)      *
INCLD V2000003 V2TEST * V2InnerClass.java          *
*
* Nested SCLM controlled jars to include          *
*
INCL V2JART1 ARCHDEF  * DateService.jar (4)        *
*
* Build script and generated outputs
*
SINC V2JARB1(1) J2EEBLD * J2EE JAR Build script    *
OUT1 *          J2EEJAR * V2TEST.jar (3)           *
LIST *          J2EELIST
```

Figure 1. Exemple d'ARCHDEF d'application Jar (JAR)

L'exemple suivant montre le script JAR correspondant.

```
<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>
```

Figure 2. Exemple de script de génération JAR J2EE

Objets de génération JAVA/J2EE générés

Les objets suivants sont générés :

- Compilation de toutes les sources Java dans des classes de sortie, stockées dans le type SCLM JAVACLAS.
- Classes enregistrées dans SCLM et noms abrégés/longs enregistrés dans les tables de conversion.
- Fichier JAR éventuellement créé (contient les classes et éventuellement d'autres composants de projet Java (comme XML, HTML, etc.) dans une structure de module).
- Objets Jar enregistrées dans SCLM et noms abrégés/longs enregistrés dans la table de conversion.
- Structure Jar déterminée par l'ARCHDEF utilisée. Les noms longs associés aux membres dans l'ARCHDEF déterminent le format de conditionnement des Jar.

- Fichier JAR EJB facultatif (contient les classes et éventuellement d'autres composants de projet Java (comme XML, HTML, JSP, etc.) dans une structure de module.
- Fichier WAR Web facultatif basé sur le fichier web.xml J2EE dans le projet J2EE et enregistré dans SCLM, comme décrit ci-dessus.
- Fichier EAR facultatif pour le déploiement basé sur le fichier application.xml dans le projet J2EE et enregistré dans SCLM, comme décrit précédemment.
- Toutes les sorties de liste sont enregistrées dans le type SCLM J2EELIST.

Traducteurs de langage pour le support JAVA/J2EE

SCLM Developer Toolkit requiert de nouveaux traducteurs de langage définis dans SCLM pour le support JAVA/J2EE. Ces traducteurs de langage sont fournis dans les membres FEK.SFEKSAMV, comme indiqué ci-dessous :

Tableau 2. Traducteurs SCLM Developer Toolkit

Traducteur	Description
BWBTRANJ	Exemple de traducteur de membre par défaut. Pas d'analyse syntaxique. Similaire à SCLM FLM@TEXT. Ce traducteur peut être personnalisé pour créer les définitions de langage J2EEPART, J2EEBIN, BINARY et TEXT.
BWBTRANS	Exemple de traducteur de langage SQLJ. LANG=SQLJ
BWBTRAN1	Exemple de traducteur de langage Java. LANG=JAVA
BWBTRAN2	Exemples de traducteur de langage JAVA/J2EE incorporant Ant (pour plusieurs compilations Java et générations JAR, WAR et EAR)
BWBTRAN3	Exemple de traducteur de langage J2EE pour le support SCLM ARCHDEF J2EE. LANG=J2EEOBJ

L'administrateur SCLM doit copier ces exemples, les renommer si nécessaire, puis les générer dans la bibliothèque PROJDEFS.LOAD de chaque projet SCLM nécessitant le support Java. Ces traducteurs doivent être ajoutés ou compilés dans la définition de projet.

Un exemple de définition de projet pour les projets JAVA/J2EE et les composants hôte est fourni dans l'exemple BWBSCLM.

Définitions de langage SCLM

Les exemples de traducteur définissent les langages suivants :

J2EEANT

Il s'agit du principal traducteur de génération pour les générations JAVA/J2EE. Ce traducteur de vérification est appelé lors de la génération d'une ARCHDEF J2EE. Il est appelé car le script de génération JAVA/J2EE, stocké dans le type SCLM J2EEBLD, est sauvegardé dans SCLM avec le langage J2EEANT. Il est ensuite référencé via le mot clé SINC dans l'ARCHDEF.

Ce traducteur de vérification détermine quels composants doivent être générés (y compris les ARCHDEF imbriquées) et, suivant les modes de génération, copie ces composants dans le répertoire WORKAREA du système de fichiers z/OS UNIX System Services. Un squelette XML Ant est personnalisé de manière dynamique en fonction du script de génération et les composants générés dans la zone de travail à l'aide de l'utilitaire Ant. Les fichiers classe sont transmis aux traducteurs de langage JAVA/JAVABIN pour enregistrer à nouveau les fichiers dans SCLM. Les

objets J2EE générés, tels que JAR, WAR ou EAR, sont transmis au traducteur de langage ARCHDEF (J2EEOBJ) pour être à nouveau enregistrés dans SCLM.

J2EEBIN

Type de langage qui spécifie un composant ASCII ou binaire JAVA/J2EE et qui est défini par l'exemple BWBTRANJ. Aucune analyse syntaxique particulière n'est effectuée lors de la génération de cette définition de langage. Les fichiers binaires JAVA/J2EE et les fichiers texte que vous souhaitez enregistrer au format ASCII peuvent être attribués de manière générique sous cette définition de langage si aucune analyse de génération particulière n'est requise.

J2EEOBJ

Ce traducteur de génération final est appelé dans le cadre du processus de génération d'ARCHDEF. Ce traducteur détermine quels objets J2EE (JAR, WAR, EAR) étaient déjà générés dans le traducteur J2EEANT et copie ces objets dans SCLM avec le nom abrégé indiqué. Ce traducteur est référencé par le mot clé LKED dans l'ARCHDEF.

J2EEPART

Type de langage qui spécifie un composant JAVA/J2EE et qui est défini par l'exemple BWBTRANJ. Aucune analyse syntaxique particulière n'est effectuée lors de la génération de cette définition de langage. Les sources autres que Java ou les composants J2EE qui requièrent une conversion de langage ASCII/EBCDIC peuvent être attribués de manière générique sous cette définition de langage si aucune analyse de génération particulière n'est requise (par exemples, html, XML, .classpath, .project ou tables de définition). La définition de langage TEXT peut éventuellement être utilisée.

JAVA Type de langage pour source Java, défini par l'exemple BWBTRAN1. Le traducteur Java détermine le type de génération effectué sur la source Java.

Remarque : Cette définition de langage doit être affectée aux programmes Java si vous souhaitez enregistrer le code source Java en EBCDIC sur l'hôte (à savoir que la source peut être affichée et modifiée directement sur l'hôte via ISPF). L'avantage de définir des programmes avec cette définition de langage est de pouvoir modifier et afficher la source directement sur l'hôte z/OS. Par contre, les pages de codes doivent être converties lors de la migration ou de l'importation de projets du client vers l'hôte.

- Scénario 1 : Génération effectuée sur un programme Java seul.

Le traducteur Java compile la source en classes de sortie. La classe est stockée dans SCLM sous le type JAVACLAS. La sortie de compilation Java est stockée dans le type JAVALIST.

Les dépendances des chemins d'accès aux classes peuvent être satisfaites en stockant des fichiers JAR dépendants dans le répertoire des chemins d'accès aux classes spécifié dans le paramètre CLASSPATH_JARS du membre \$GLOBAL. Pour plus d'informations, voir «\$GLOBAL», à la page 24.

- Scénario 2 : Une génération au niveau d'ARCHDEF (ARCHDEF appelle le script de génération J2EEANT référencé par le mot clé SINC) laisse le script Ant spécifié

effectuer la génération. Le traducteur Java, lorsqu'il est appelé par l'ARCHDEF, se contente de copier les classes de sortie dans SCLM. Un fichier récapitulatif de la génération ANT est stocké dans JAVALIST. Chaque composant Java possède une table stockée dans JAVALIST.

JAVABIN

Type de langage similaire à Java et utilisé lors du stockage de la source Java au format ASCII dans SCLM.

SQLJ Type de langage du code source SQLJ défini par l'exemple BWBTRAN1. Les membres SQLJ définis avec ce traducteur de langage appellent le traducteur de langage SQLJ au moment de la phase de génération. La source SQLJ est convertie en source Java, puis compilée dans des classes et des objets sérialisés (fichiers .ser) dans le type SQLJSER. Les membres DBRM peuvent aussi être générés dans le type DBRMLIB.

Remarque : Les composants source internes zippés de tous les objets tels que les fichiers JAR, WAR et EAR sont au format ASCII pour que ces objets puissent être distribués à toutes les plateformes.

Fichiers SCLM pour JAVA/J2EE

Il est recommandé de créer des fichiers source SCLM cible de RECFM=VB, LRECL=1024 pour toute source JAVA/J2EE à stocker dans SCLM à partir du client Toolkit afin d'autoriser les enregistrements de type long.

Les éditeurs du client Eclipse créent des fichiers de longueur d'enregistrement variable et, pour conserver l'intégrité, les fichiers cible de l'hôte dans SCLM doivent également être au format RECFM=VB. Avec l'utilisation de fichiers de longueur d'enregistrement fixe (RECFM=FB), les membres importés contiennent des espaces à la fin de l'enregistrement.

Types SCLM

Un certain nombre de types SCLM doivent être créés pour le support JAVA/J2EE. Certains de ces types sont obligatoires et doivent être créés pour que le support JAVA/J2EE fonctionne.

Attributs de fichier recommandés pour certains types standard

Les attributs de fichier par défaut DSORG=PO TRACKS(1,5) DIR=50 BLKSIZE=0 (déterminés par le système) sont recommandés pour les TYPES SCLM ci-après.

En outre, le format d'enregistrement et les attributs de longueur d'enregistrement suivants sont recommandés :

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

Type SCLM	RECFM	LRECL
J2EEEAR	VB	256
J2EEJAR	VB	256
J2EEWAR	VB	256
SQLJSER	VB	256
Autres types de fichier source pour Java/J2EE	VB	1024

ARCHDEF

Le type ARCHDEF contient les membres d'ARCHDEF JAVA/J2EE.

Les parties correspondant au nom long de chaque membre ARCHDEF définissent la structure des projets JAVA/J2EE. L'ARCHDEF d'un projet donné peut être créé de manière dynamique à partir du client lors de la migration vers de nouveaux projets ou mis à jour lors de l'ajout de nouveaux composants à un projet existant.

L'ARCHDEF SCLM est le fichier SCLM principal pour la définition des éléments d'un projet JAVA/J2EE. Dans le cas des applications JAVA/J2EE, l'ARCHDEF représente la manière dont l'application J2EE est structurée dans l'espace de travail du projet IDE client.

La structure des fichiers de projet de l'application est répliquée dans l'ARCHDEF (à l'aide du nom abrégé de l'hôte SCLM pour mapper la structure du nom long). D'autres mots clés dans l'ARCHDEF, tels que LINK, SINC et OUT1 indiquent à SCLM la nature J2EE de ce projet et le code source inclut un script de génération JAVA/J2EE pour simplifier le traitement de la génération de ce projet.

J2EEBLD

Le type J2EEBLD est requis pour les processus de génération et de déploiement Java et J2EE et contient :

- Les scripts de génération J2EEBLD permettant de piloter les processus de génération et de déploiement Ant.
- Les scripts ANTXML Java et J2EE à appeler pour les générations et les déploiements.
- \$GLOBAL, qui spécifie les propriétés par défaut du projet SCLM pour le processus de génération JAVA/J2EE. Pour plus d'informations, voir «Format \$GLOBAL», à la page 9 et «\$GLOBAL», à la page 24.

Remarque : Les exemples de scripts ANTXML Java et J2EE sont fournis. Généralement, ces scripts nécessitent peu ou pas de personnalisation de la part de l'utilisateur. Les variables dépendant de l'utilisateur et du site sont personnalisées dans les scripts J2EEBLD pour remplacer les variables ANTXML par défaut. Pour plus d'informations, voir «Squelettes de génération XML Ant JAVA/J2EE», à la page 16.

JAVALLIST

Le type JAVALLIST est requis pour le processus de génération Java et contient les sorties de liste des générations Java.

J2EELIST

Le type J2EELIST est requis pour le processus de génération J2EE et contient les sorties sous forme de liste des générations J2EE.

DBRMLIB

Il s'agit techniquement d'un type DB2.

DBRMLIB est requis pour le support SQLJ et stocke les modules d'interrogation de base de données.

JAVACLAS

Le type JAVACLAS est requis pour les processus de génération Java et J2EE et contient les fichiers de classe de sortie des générations associées aux définitions de langage JAVA, J2EEANT.

J2EEEAR

Le type J2EEEAR est requis pour le processus de génération J2EE et contient la sortie EAR des générations associées à la définition de langage J2EEANT.

J2EEJAR

Le type J2EEJAR est requis pour les générations JAVA/J2EE et contient la sortie JAR des générations associées à la définition de langage J2EEANT.

J2EEWAR

Le type J2EEWAR est requis pour le processus de génération J2EE et contient la sortie WAR des générations associées à la définition de langage J2EEANT.

SQLJSER

Le type SQLJSER est requis pour la procédure de génération J2EE/SQLJ et stocke les profils sérialisés SQLJ.

Types <Java/J2EE>

Un type SCLM différent est requis pour chaque projet JAVA/J2EE à enregistrer dans SCLM. Ceci permet d'éviter que des conflits de fichiers portant le même nom se produisent avec les projets JAVA/J2EE. Pour plus d'informations, voir «Mappage de projets J2EE à SCLM», à la page 18.

Formats des membres SCLM

Cette section décrit les formats des membres SCLM.

Format \$GLOBAL

Le format \$GLOBAL est de type J2EEBLD et possède le langage J2EEPART. Il doit utiliser le nom \$GLOBAL et les variables sont définies dans un format de langage balisé.

\$GLOBAL spécifie les propriétés par défaut du projet SCLM pour le processus de génération JAVA/J2EE. Il doit être stocké dans le type SCLM J2EEBLD.

Pour une description détaillée des paramètres des membres \$GLOBAL, voir «\$GLOBAL», à la page 24.

Reportez-vous à l'exemple de code suivant :

```
<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"/>
```

Figure 3. \$GLOBAL - Variables d'environnement SCLMDT

Format d'ARCHDEF J2EE

Le format d'ARCHDEF J2EE a le type ARCHDEF et le langage ARCHDEF.

L'ARCHDEF utilise les mots clés d'architecture SCLM standard pour indiquer à SCLM comment traiter la génération de l'ARCHDEF.

```
LKED J2EEOBJ
INCLD SourceFile SourceType
INCL ArchdefName ArchdefType
SINC BuildScriptname J2EEBLD
OUT1 * J2EEOutputObjectType
LIST * J2EELIST
```

LKED

Indique qu'il s'agit d'une ARCHDEF LEC et spécifie le traducteur de langage ARCHDEF à appeler (toujours J2EEOBJ pour les ARCHDEF J2EE).

INCLD

Inclusion SCLM du composant J2EE. *SourceFile* correspond au nom du membre source (par exemple, source Java) inclus dans cette ARCHDEF. *SourceType* correspond au type SCLM qui contient le membre. Dans une ARCHDEF générée par SCLM Developer Toolkit, un commentaire indique le nom complet du fichier, tel qu'il existait dans le projet IDE sur le plan de travail.

INCL Inclusion SCLM d'une autre ARCHDEF imbriquée, comme l'ARCHDEF qui contient le manifeste d'une application EJB.

SINC Inclusion source du script de génération J2EEBLD. *BuildScriptName* correspond au nom du script de génération. Le type de source est toujours J2EEBLD.

OUT1 Indique le type d'objet J2EE créé par cette ARCHDEF. Le nom du membre est toujours *. *J2EEOutputObjectType* a la valeur J2EEEAR, J2EEWAR ou J2EEJAR. Le membre créé recevra un nom du type de nom abrégé du fichier JAR, WAR ou EAR.

LIST Liste récapitulative des composants et de l'audit de la génération de l'ARCHDEF. Le nom du membre est toujours *. Le type de source est toujours J2EELIST. Le membre se voit attribuer un nom de la même valeur que le nom de membre de l'ARCHDEF.

Exemples d'ARCHDEF J2EE

L'exemple suivant montre le code JAR :

```
*
* Initially generated on 10/05/2006 by SCLM DT V2
*
LKED  J2EE0BJ          * J2EE Build translator
*
* Source to include in build
*
INCLD AN000002 V2TEST  * com/Angelina.java          *
INCLD V2000002 V2TEST  * com/V2Java1.java          *
INCLD V2000003 V2TEST  * V2InnerClass.java          *
*
* Build script and generated outputs
*
SINC  V2JARB1 J2EEBLD  * J2EE JAR Build script      *
OUT1  *        J2EEJAR * V2TEST.jar                *
LIST  *          J2EELIST
```

Figure 4. Exemple d'ARCHDEF d'application Jar (JAR)

L'exemple suivant montre le code WAR :

```
*
* Initially generated on 5 Sep 2006 by SCLM DT V2
*
LKED  J2EE0BJ          * J2EE Build translator
*
* Source to include in build
*
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           *
*
* Build script and generated outputs
*
SINC  SAMPLE5 J2EEBLD  * J2EE WAR Build script      *
OUT1  *        J2EEWAR * Sample5.war                *
LIST  *          J2EELIST
```

Figure 5. Exemple d'ARCHDEF d'application Web (WAR)

L'exemple suivant montre le code 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 * J2EE EJB JAR Build script *
OUT1  *      J2EEJAR * DateService.jar *
LIST  *      J2EELIST

```

Figure 6. Exemple d'ARCHDEF d'application EJB (EJB)

L'exemple suivant montre le code EAR :

```

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

```

Figure 7. Exemple d'application EAR (EAR) ARCHDEF

Format du script de génération Ant J2EE

Le format du script de génération Ant J2EE a le type J2EEBLD et le langage J2EEANT. Il peut s'agir d'un nom comportant jusqu'à huit caractères dont les variables sont définies dans un format de langage balisé. Les scripts de génération des fichiers JAR, WAR et EAR sont très similaires. Un exemple de syntaxe de script de génération WAR est indiqué ci-dessous. Pour les fichiers JAR et EAR, les scripts de génération, les variables sont identiques, à l'exception de WAR_NAME que vous devez remplacer par JAR_NAME et WAR_NAME.

Reportez-vous à l'exemple de script de génération suivant :

```
<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"/>

  <!-- WAR file name to be created by this build process -->
  <!-- include suffix of .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>
```

Figure 8. Script de génération Ant J2EE

Les scripts de génération SCLM remplacent les variables définies par le client de manière dynamique lors de l'exécution du script de génération Ant. Ces variables possèdent les valeurs indiquées dans le tableau 3.

Tableau 3. Variables définies par le client

Variable	Description
Nom du projet J2EE	Type de projet Java/J2EE généré. Ce nom de projet temporaire est défini dans le script de génération pour que l'utilitaire Ant puisse l'utiliser pendant la génération. Les valeurs suivantes seront définies : <ul style="list-style-type: none">• J2EE EAR Project• J2EE WAR Project• J2EE EJB Project• Projet JAVA Cette variable n'a pas besoin d'être personnalisée
SCLM_ARCHDEF	Nom de l'ARCHDEF ou de l'ARCHDEF générée
SCLM_ANTXML	Nom du squelette XML Ant à utiliser pour la génération

Tableau 3. Variables définies par le client (suite)

Variable	Description
SCLM_BLDMAP	Valeur Oui ou Non. Si Oui, inclure la mappe de génération SCLM dans le répertoire MANIFEST dans JAR, WAR ou EAR. Fournit l'audit et la mappe de génération des composants inclus.
JAVA_SOURCE	Valeur : Yes ou No. Si la valeur est Yes, incluez la source Java dans JAR, WAR ou EAR.
CLASSPATH_JARS	Répertoire de chemin de classes z/OS UNIX System Services utilisé pour résoudre les dépendances de chemin de classes lors de la génération. Tous les fichiers jar situés dans ce répertoire seront utilisés dans le chemin de classe.
CLASSPATH_JARS_FILES	Noms des fichiers classe et JAR à inclure individuellement dans la génération. Il peut s'agir d'une liste, comme dans l'exemple suivant : <code><property name="CLASSPATH_JARS_FILES" value="V2J4.jar,V2J3.jar" /></code>
ENCODING	Page de codes ASCII ou EBCDIC pour JAVA. Il s'agit de la source JAVA de la page de codes enregistrée sur l'hôte z/OS. Par exemple : <ul style="list-style-type: none"> • Pour le format ASCII JAVA, la page de codes standard doit être ISO8859-1 • Pour le format EBCDIC JAVA, la page de codes standard doit être IBM-1047
JAR_FILE_NAME EJB_NAME WAR_NAME EAR_NAME	Nom du fichier JAR, EJB JAR, WAR ou EAR.
DEBUG_MODE	Spécifiez 'on' pour forcer Developer Toolkit à ne pas supprimer les fichiers de génération du répertoire WORKAREA. Cela est utile si vous devez vérifier la structure d'une Java/J2EE générée.

Dépendances CLASSPATH

La source Java d'une ARCHDEF peut comporter des dépendances de chemin de classes sur d'autres bibliothèques ou fichiers classe Java. Si ces dépendances sont situées sur des composants Java contenus dans la même structure d'ARCHDEF, elles sont résolues lors de la génération d'ARCHDEF (en mode conditionnel ou forcé).

Toutefois, un composant d'ARCHDEF J2EE peut comporter des dépendances de chemin de classe sur des fichiers JAR externes ou des membres contenus dans d'autres ARCHDEF. Dans ce cas, le script de génération J2EE associé à l'ARCHDEF peut contrôler les dépendances de chemin de classes à l'aide des mots clés suivants :

CLASSPATH_JARS

Il s'agit du nom de répertoire dans le système de fichiers z/OS UNIX System Services susceptible de contenir tous les fichiers JAR dépendants externes et les fichiers de classe de cette génération d'ARCHDEF.

Ce répertoire peut être mis à jour avec les fichiers CLASSPATH via la fonction d'équipe client Charger les fichiers JAR qui permet de copier les fichiers JAR du client dans le répertoire classpath. La fonction Copier le fichier de SCLM vers le classpath qui permet de copier les fichiers JAR enregistrés dans SCLM dans le répertoire classpath est également disponible.

CLASSPATH_JARS_FILES

Ce mot clé peut être utilisé pour sélectionner séparément des fichiers classe ou JAR à utiliser dans le classpath. Si ce mot clé est utilisé, les fichiers de classe/JAR répertoriés sont extraits de SCLM ou s'ils ne sont pas dans SCLM, le répertoire référencé par la variable CLASSPATH_JARS est

recherché pour être extrait. Si ce mot clé est utilisé, seuls ces fichiers répertoriés sont utilisés dans le classpath.

Exemples de script J2EE

L'exemple suivant montre le 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>
```

Figure 9. Exemple de script de génération JAR J2EE

L'exemple suivant montre le 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"/>
<!-- WAR file name to be created by this build process -->
<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"/>
</path>
<fileset dir="${CLASSPATH_JARS}" includes="**/*.jar, **/*.zip"/>
</ANTXML>
```

Figure 10. Exemple de script de génération WAR J2EE

L'exemple suivant montre le 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>
```

Figure 11. Exemple de script de génération EJB J2EE

L'exemple suivant montre le 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>
```

Figure 12. Exemple de script de génération EAR J2EE

Squelettes de génération XML Ant JAVA/J2EE

Cette section répertorie les exemples de squelette de génération Ant fournis dans la bibliothèque FEK.SFEKSAMV. Ces exemples de membre peuvent être copiés dans le type SCLM J2EEBLD de la hiérarchie SCLM pour être référencé et utilisé par les scripts de génération JAVA/J2EE. Ces scripts sont des fichiers de variable de propriétés qui écrasent les fichiers de squelette XML Ant.

Les exemples de squelette de génération J2EE fournis pour générer un simple fichier JAR, un projet SQLJ, un JAR d'EJB, un WAR ou un EAR ou pour le déploiement peuvent généralement être utilisés tels quels, sans personnalisation

par l'utilisateur. Sachez toutefois que certains projets J2EE peuvent ne pas s'appliquer au modèle standard, auquel cas les squelettes XML Ant fournis risquent de devoir être personnalisés.

Remarque : Les scripts de génération JAVA/J2EE peuvent être générés via l'application client SCLM Developer Toolkit. Ces scripts de génération utilisent un squelette XML Ant référencé (voir ci-après) et une ARCHDEF dans la procédure de génération JAVA/J2EE.

Une description détaillée des scripts de génération, des squelettes Ant et des exemples de traitement de génération JAVA/J2EE sont fournis dans le Guide d'utilisation de SCLM Developer Toolkit livré avec le plug-in du client.

BWBJAVAA

Exemple de squelette de génération JAVA XML Ant

Ce squelette Ant est utilisé par un script de génération Java pour compiler plusieurs programmes Java et éventuellement créer un fichier JAR (Java Archive) qui possède une structure déterminée par une ARCHDEF indiquée.

BWBEJBA

Exemple de squelette de génération d'EJB J2EE XML Ant

Ce squelette Ant est utilisé par un script de génération J2EE pour compiler ou générer un projet EJB qui crée généralement un JAR EJB qui possède une structure déterminée par une ARCHDEF spécifiée.

BWBWEBA

Exemple de squelette de génération WEB J2EE XML Ant

Ce squelette est utilisé par un script de génération J2EE pour compiler ou générer un projet WEB qui crée généralement un fichier d'archive WEB (WAR).

BWBEARA

Exemple de squelette d'assemblage d'EAR J2EE XML Ant

Ce squelette est utilisé par un script de génération J2EE comme processus d'assemblage en préparation du déploiement d'applications J2EE. Ce processus génère des fichiers EAR (Enterprise Archive) qui peuvent être déployés sur un serveur d'applications Web, tel qu'un serveur d'applications WebSphere.

BWBSQLB

Exemple de script de génération Java/SQLJ

Ce squelette est utilisé par un script de génération J2EE pour compiler ou générer un projet JAR qui utilise SQLJ.

BWBSQLBE

Exemple de script de génération EJB/SQLJ

Ce squelette est utilisé par un script de génération J2EE pour compiler ou générer un projet EJB qui utilise SQLJ.

BWBC9DTJ

Exemple de conversion de Cloud 9 vers SCLM DT.

BWBDEPJ1

Exemple de mise à jour de fichiers .ser SQLJ au sein d'un fichier JAR lors du déploiement à l'aide de db2sqljcustomize.

BWBDEPJ2

Exemple conçu pour db2sqljcustomize dans lequel la propriété longname copie le fichier JAR indiqué du groupe spécifié et les emplacements de types de SCLM dans le répertoire de destination défini par "dest".

BWBDEPJ3

Cet exemple de routine personnalise les fichiers .ser stockés dans les fichiers JAR sélectionnés pour un déploiement à l'aide de db2sqljcustomize.

BWBTRANX

Exemple de convertisseur de génération SCLM pour les messages d'erreur de génération SYSXML pour COBOL.

Mappage de projets J2EE à SCLM

IBM SCLM Developer Toolkit permet de gérer, de générer et de déployer des projets dans SCLM. Cette section explique comment configurer la structure de projets SCLM pour prendre en charge le développement d'applications distribuées, tel que JAVA/J2EE.

De nombreux projets JAVA/J2EE entraînent la création d'un fichier EAR exécutable. Cette application est un assemblage de projets, généralement des EJB et des applications et, dans les environnements IDE, ces projets sont généralement développés comme structures de projets individuelles associées à un projet EAR.

Cette forme de structures de projets multiples ne se mappe pas directement à SCLM. En effet, un projet SCLM ne peut pas être associé à un autre projet SCLM pour fournir une forme de structure de projets agrégée. Toutefois, SCLM offre un moyen permettant de prendre en charge cette structure IDE à projets multiples dans un même projet SCLM, à l'aide des types.

Les projets SCLM peuvent être définis avec plusieurs types de source. Chaque type peut contenir un seul projet IDE. Si nous tentions d'enregistrer plusieurs projets IDE Eclipse dans SCLM sans une certaine forme de séparation, chaque fichier .classpath et .project du projet serait remplacé car chaque fichier a été ajouté à SCLM. L'utilisation de différents types de source permet à ces fichiers et à tous les autres fichiers associés au projet, d'être enregistrés dans SCLM en toute sécurité.

Avec ce mappage, les projets IDE seraient enregistrés indépendamment dans SCLM avec le type comme principal élément de différenciation. Par exemple : EJB1 est enregistré dans le projet SCLM SCLMPRJ1 sous le type EJB1. En utilisant cette structure, il est possible de mapper la structure de projets IDE à des types indépendants dans le projet SCLM.

Remarques :

1. Il n'est pas nécessaire de mapper un nom de projet dans l'IDE au nom de type SCLM ; ces noms existent indépendamment des autres.
2. Les noms de type sont restreints à huit caractères ; un projet IDE appelé 'ProjetDeux' ne peut donc pas posséder le nom de type correspondant 'ProjetDeux'. Vous pouvez utiliser 'Proj2' à la place.

Il est donc important que la structure de projet SCLM soit conçue pour permettre le mappage de différents projets IDE dans sa structure. En effet, dans les grands projets SCLM, il peut s'avérer utile d'ajouter des types de projet supplémentaires dans la mesure où cette opération requiert une modification de la définition du projet SCLM, une régénération de la définition de projet SCLM et l'allocation de fichiers pour les nouveaux types.

Cette structure ne se limite pas aux projets de style J2EE, mais peut s'appliquer à toutes les situations dans lesquelles plusieurs projets sont développés, procurant une certaine forme de dépendance entre eux.

Recommandations de mappage des projets J2EE à SCLM

La liste suivante offre des recommandations permettant de mapper des projets J2EE (et autres) à SCLM :

- Identifiez la composition du projet J2EE en termes de fichiers EJB, d'applications Web et autres, pour que cette composition puisse être utilisée pour planifier la structure de projet SCLM.
- Pour chaque composant du projet IDE J2EE, créez un type correspondant dans le projet SCLM. Il est utile de déterminer un type de convention de dénomination explicite pour cela. Vous pouvez nommer les projets IDE indépendamment du type SCLM, mais une certaine corrélation facilitera l'administration.
- Dans la mesure où les exigences relatives au projet peuvent changer, créez d'autres définitions de type afin de permettre l'ajout d'autres composants, comme des EJB. L'anticipation de services supplémentaires peut être prévue dans la structure de type.
- Le mappage de plusieurs projets IDE dans un seul et même projet SCLM est pris en charge par la construction de type. Il est également utile d'appliquer une structure de regroupement prenant en compte la définition de type de ce projet.
- Des conventions de regroupement de type Java peuvent également être définies au niveau du projet afin d'éviter le risque de conflits de dénomination de fichiers source.
- Si l'environnement IDE comporte plusieurs projets, il est conseillé de répliquer cette structure dans SCLM à l'aide du type utilisant SCLM.

L'utilisation de plusieurs types SCLM pour enregistrer des projets IDE est également liée à l'opération de la structure ARCHDEF pour la génération de ces projets IDE.

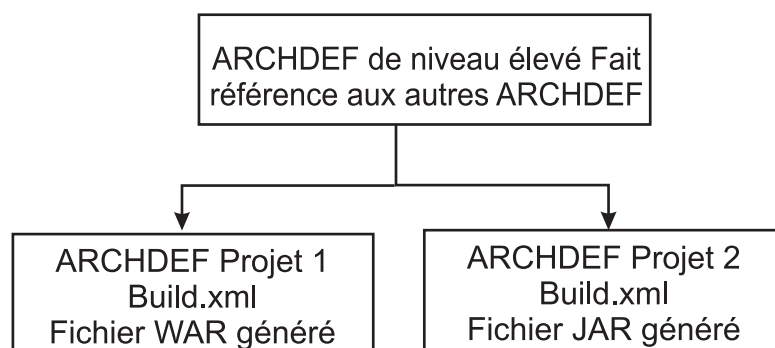


Figure 13. Hiérarchie de génération SCLM

Le fichier ARCHDEF contient la liste des membres qui constituent une génération. Dans un contexte J2EE, une génération peut entraîner la constitution de plusieurs fichiers WAR et JAR dans un fichier EAR. Cette séparation de projets est similaire à la structure de type qui définit le projet dans SCLM. En disposant d'une ARCHDEF de haut niveau qui fait référence aux composants qui constituent la génération, il est possible d'obtenir un environnement de génération structuré. Ceci dépend de la définition réelle de la structure de projet lorsque vous définissez les types dans SCLM.

La définition du projet d'une manière structurée permet également :

- La migration des fichiers d'un type de projet SCLM ou ARCHDEF vers un projet IDE sans avoir à connaître chaque composant.
- La structure ARCHDEF basée sur la définition de type permet également de mapper les dépendances de projet de manière plus efficace. Il est en effet courant que les projets IDE fassent référence à d'autres projets IDE dans l'espace de travail. Ceci est possible grâce à l'utilisation du mot clé SCLM INCL dans les ARCHDEF, qui permet d'inclure d'autres projets IDE, référencés par d'autres ARCHDEF, en imbriquant les ARCHDEF dans des ARCHDEF de niveau supérieur.

Lors de la génération d'autres applications ayant des références ou des dépendances sur d'autres objets de génération, comme des fichiers JAR, d'autres projets ou d'autres classes, plusieurs approches sont possibles :

1. Incluez la référence au fichier JAR à l'aide de l'instruction INCLD dans l'ARCHDEF. Cette opération permet de générer l'application avec la référence de bibliothèque dans le package de génération final.
2. Incluez le projet IDE en tant qu'ARCHDEF de projet SCLM INCL imbriqué.
3. Incluez le fichier JAR dépendant dans le répertoire CLASSPATH.
 - Si le projet IDE se rapporte à un fichier JAR, mais qu'il n'est pas destiné à faire partie du package de génération final, le fichier de bibliothèque peut être copié dans la CLASSPATH système à l'aide de fonction Charger les fichiers JAR dans Developer Toolkit. Cette opération aura pour effet de rendre ce service disponible depuis un autre projet SCLM. Au moment de la génération, le projet IDE qui fait référence au fichier JAR est résolu. Toutefois, pendant l'exécution, le fichier JAR doit être disponible dans une instruction PATH.
 - Faites référence à un fichier JAR qui réside dans le même projet SCLM en utilisant la propriété CLASSPATH_JARS_FILES dans le script de génération.

Déploiement de SCLM Developer Toolkit

SCLM Developer Toolkit offre plusieurs fonctions de déploiement. Vous pouvez déployer des fichiers d'archive d'entreprise (EAR) dans un serveur WebSphere Application Server (WAS). D'autre part, tous les composants générés ou contrôlés par SCLM Developer Toolkit peuvent être distribués à l'aide d'un script de déploiement personnalisable. Des exemples de script permettant de copier un fichier EAR vers un hôte distant à l'aide des commandes secure copy (SCP) et secure FTP (SFTP) sont fournis.

Il existe principalement deux scripts au coeur du déploiement ; le premier type de script, celui modifié par l'utilisateur, est le script des propriétés, qui contient une liste de paramètres pour l'opération de déploiement. Le second est le script d'action contenant les étapes requises pour exécuter l'opération de déploiement.

Le déploiement est lancé à partir du plug-in client de SCLM Developer Toolkit et le type de déploiement est choisi en appuyant sur le bouton approprié de l'écran de déploiement. Le choix de l'action de déploiement a un impact sur le contenu du script des propriétés. La plupart des scripts incluent la propriété SCLM_ANTXML, qui contient le nom de membre du script d'action correspondant. Developer Toolkit sélectionne le script de propriétés généré et le superpose au script d'action, avant d'appeler le script d'action résultant.

Vous trouverez ci-après une liste d'exemples de script d'action de déploiement Ant qui sont fournis dans la bibliothèque FEK.SFEKSAMV. Ces exemples de membre peuvent être copiés dans le type SCLM J2EEBLD de la hiérarchie SCLM pour être référencé et utilisé par les scripts de propriétés générés. Ces scripts sont des fichiers de variable de propriétés qui écrasent les scripts d'action de déploiement XML Ant référencés ci-après. Ces scripts doivent être stockés avec un langage de type texte, tel que TEXT ou J2EEPART.

Membre	Description
BWBDEPLA	Déploiement EAR WAS.
BWBRDEPL	Déploiement EAR WAS distant.
BWBSCOPY	Déploiement Secure copy. Copie un objet de génération d'un hôte vers un autre à l'aide de SCP.
BWBSFTP	Déploiement Secure FTP. Copie un objet de génération d'un hôte vers un autre à l'aide de SFTP.

Pour que ces scripts de génération puissent être utilisables à partir de plusieurs groupes, l'administrateur doit générer les scripts et les promouvoir au niveau de groupe le plus élevé disponible dans le projet.

Le traitement diffère légèrement suivant les types de script générés.

Déploiement de WebSphere Application Server (WAS)

Pour le déploiement de WebSphere Application Server (WAS) la propriété SCLM_ANTXML ne fait pas référence à un script d'action Ant, mais à un script d'action JACL. Vous pouvez également utiliser l'outil wsadmin fourni avec WAS sous z/OS.

Cet outil requiert un script JACL pour assister la procédure de déploiement. Si cette méthode de déploiement est utilisée, le script JACL doit être créé sous la forme d'un fichier ASCII dans un répertoire z/OS UNIX pour que la procédure de déploiement puisse être appelée.

La personnalisation de Developer for System z fournit l'exemple de script JACL (ASCII) /etc/rdz/sclmdt/CONFIG/scripts/deploy.jacl (/etc/SCLMDT représentant le répertoire etc par défaut de SCLM Developer Toolkit).

Vous trouverez des exemples JACL supplémentaires dans la documentation de WebSphere Application Server (WAS).

Les emplacements de répertoire de l'outil wsadmin (wsadmin.sh) et du script JACL (deploy.jacl) peuvent être configurés dans la page des préférences, sous **Equipe > Préférences du SCLM > Options de script de génération**. Le client SCLMDT permet de générer un script de déploiement qui peut ensuite servir de base de

génération. (La procédure de déploiement est déclenchée par une demande de fonction de déploiement au script de déploiement qui est stocké dans le type SCLM J2EEBLD).

Les exemples de script d'action à stocker dans le type SCLM J2EEBLD pour le déploiement de WAS ou le déploiement à distance de WAS sont BWBDEPLA et BWBRDEPL.

Déploiement de SCLM sur UNIX System Services

SCLM Developer Toolkit offre un moyen de déployer les fichiers stockés dans le référentiel SCLM sur le système de fichiers z/OS UNIX System Services de la même partition logique. Il est ainsi facile de déployer un objet généré par SCLM dans un environnement où il peut être exécuté ou même déployé sur un hôte éloigné à l'aide du déploiement sécurisé décrit ci-après.

Il n'existe pas de script d'action pour cette action. Sélectionnez les membres de SCLM et générez le script de propriétés requis à l'aide du bouton Inclure les membres SCLM. Les fichiers sont copiés de l'emplacement SCLM sélectionné vers un répertoire spécifié sur le système de fichiers z/OS UNIX System Services. Si ce répertoire n'existe pas, une erreur se produit.

Déploiement sécurisé

Cette option permet de copier des objets déployables sur un hôte éloigné à l'aide des commandes secure copy (SCP) et secure FTP (SFTP). En combinant le script des propriétés de déploiement sécurisé et l'inclusion des membres SCLM, les fichiers requis peuvent être sélectionnés dans la hiérarchie SCLM, copiés dans un emplacement du système de fichiers z/OS UNIX System Services, puis copiés sur la machine de destination à partir de cet emplacement du système de fichiers z/OS UNIX System Services à l'aide des commandes SCP (copie sécurisée) et SFTP (FTP sécurisé).

Les exemples de script d'action à stocker dans le type SCLM J2EEBLD pour le déploiement sécurisé sont BWBSCOPY et BWBSFTP.

Remarque : IBM Ported Tools for z/OS devra être commandé, installé et configuré pour la prise en charge du déploiement sécurisé. Pour savoir comment vous procurer Ported Tools, voir *IBM Rational Developer for System z Host Planning Guide* (GI11-8296). L'installation et la personnalisation de ce produit ne sont pas décrites dans le présent ouvrage.

IBM Ported Tools for z/OS fournit les protocoles suivants :

- scp pour copier des fichiers entre des réseaux. Il remplace rcp.
- sftp pour les transferts de fichier via un transport ssh chiffré. Il s'agit d'un programme de transfert de fichier similaire à ftp.

Authentification de clé publique

L'authentification de la clé publique remplace la connexion interactive et peut être automatisée dans le cadre de l'opération de déploiement sécurisé de Developer Toolkit.

Pour que l'authentification de la clé publique fonctionne comme prévu, vous pouvez utiliser un ID utilisateur de substitution pour le déploiement ou configurer chaque utilisateur que vous souhaitez autoriser à effectuer des opérations de déploiement.

Pour obtenir des instructions de configuration de l'authentification basée sur des clés via ssh-agent et de ssh-add, reportez-vous au document *IBM Ported Tools for z/OS User's Guide*. Pour plus d'informations sur l'utilisation des ID utilisateur de substitution, voir Chapitre 4, «Sécurité SCLM», à la page 47.

Autres options de déploiement

Il est également possible de créer des scripts Ant pour effectuer le déploiement, de différentes manières. Dans vos scripts, vous pouvez utiliser la balise Ant `<exec>` pour appeler un programme disponible dans le système de fichiers z/OS UNIX System Services. À l'aide de cette méthode, les scripts de génération peuvent appeler d'autres programmes, tels que FTP, pour effectuer le déploiement. Pour plus d'informations sur la création de scripts Ant, reportez-vous à la documentation Ant en ligne, à l'adresse suivante : <http://ant.apache.org/>.

Options de stockage ASCII ou EBCDIC

Les fichiers source transférés à partir du plug-in SCLM Developer Toolkit peuvent être stockés dans SCLM au format ASCII ou EBCDIC.

Généralement, toutes les sources de SCLM sont stockées au format EBCDIC afin de pouvoir être affichées et modifiées directement à partir d'ISPF/SCLM sous z/OS. Si vous ne souhaitez pas parcourir ou modifier le code directement à partir de l'hôte, vous pouvez le stocker directement (le transférer en binaire) là où la source sera stockée dans SCLM, en utilisant la page de codes ASCII/UNICODE d'origine du client. Cette méthode a l'avantage d'améliorer les performances pour les projets de grande taille stockés et importés à partir de SCLM et pour les générations JAVA/J2EE car aucune conversion d'ASCII en EBCDIC n'est effectuée.

SCLM Developer Toolkit détermine si un fichier est transféré en binaire ou si une conversion d'ASCII en EBCDIC est effectuée, en vérifiant le langage SCLM associé à chaque fichier ou membre. SCLM Developer Toolkit vérifie ensuite que le langage SCLM contient une entrée dans le fichier `TRANSLATE.conf` avec un mot clé `TRANLANG`.

Traducteurs de langage ASCII/EBCDIC

Tableau 4. Traducteurs de langage SCLM et ASCII/EBCDIC

Traducteur de langage SCLM	Description
JAVA	Membres de source Java stockés au format EBCDIC. Créés à l'aide de l'exemple BWBTRAN1.
SQLJ	Membres SQLJ stockés au format EBCDIC. Créés à l'aide de l'exemple (BWBTRANS).
JAVABIN	Membres de source Java stockés au format ASCII. Créés à l'aide de l'exemple BWBTRAN1.
J2EEPART	Fichiers J2EE pour lesquels aucune analyse syntaxique n'est requise, stockés au format EBCDIC. Créés à l'aide de l'exemple BWBTRAN1.
J2EEBIN	Fichiers J2EE pour lesquels aucune analyse syntaxique n'est requise, stockés comme fichiers ASCII ou binaires. Créés à l'aide de l'exemple BWBTRAN1.
SQLJ	Membres de source SQLJ stockés au format EBCDIC. Créés à l'aide de l'exemple BWBTRANS.
SQLJBIN	Membres de source SQLJ stockés au format ASCII. Créés à l'aide de l'exemple BWBTRANS.

Tableau 4. Traducteurs de langage SCLM et ASCII/EBCDIC (suite)

Traducteur de langage SCLM	Description
TEXT	Traducteur de texte par défaut pour lesquels aucune analyse syntaxique n'est requise, stockés au format EBCDIC. Créés à l'aide de l'exemple BWBTRAN1.
BINARY	Traducteur de langage binaire par défaut pour lequel aucune analyse syntaxique n'est requise. Créés à l'aide de l'exemple BWBTRAN1.

Par défaut, une conversion ASCII/EBCDIC est effectuée. Cela signifie que les fichiers consultés et modifiés dans le plug-in Eclipse peuvent également l'être directement sur l'hôte à partir d'ISPF/SCLM.

Il est recommandé d'utiliser le format ASCII (transfert en binaire) pour les performances de génération et de migration ou d'importation de projets, car les fichiers n'ont pas besoin d'être convertis. Cette méthode ne convient que si l'édition dans ISPF/SCLM n'est pas requise.

Suivant le traducteur de langage SCLM utilisé, le source peut être généré en ASCII ou EBCDIC.

Pour les possibilités d'utilisation entre des plateformes différentes, tous les fichiers déployables, tels que les fichiers JAR, WAR et EAR, sont générés afin que tous les objets contenus soient de type ASCII, que le source soit stocké au format EBCDIC ou non.

Remarque sur la génération JAVA/J2EE : Si la source Java est stockée en ASCII, le script de génération doit spécifier la page de codes ASCII à l'aide de la variable de propriété ENCODING pour compiler correctement la source Java.

Par exemple :

```
<property name="ENCODING" value="IS08859-1"/>
```

Le script Ant appelé utilise la commande Java avec le paramètre ENCODING=IS08859-1 pour compiler le source ASCII. La page de codes ENCODING par défaut est la page de codes EBCDIC IBM-1047.

\$GLOBAL

Dans le cadre du processus de génération JAVA/J2EE, des informations supplémentaires sont nécessaires pour exécuter les générations. Etant donné que les générations sont effectuées dans le système z/OS UNIX System Services, des informations, telles que l'emplacement du produit Java, l'emplacement du produit Ant et l'emplacement des fichiers de configuration SCLM Developer Toolkit et la zone de travail sont requises.

Vous pouvez également être amené à utiliser des versions différentes d'Ant ou de Java pour différents groupes de développement SCLM. C'est pourquoi le membre \$GLOBAL peut être propre à un groupe. Les variables d'environnement définies dans le membre \$GLOBAL peuvent être remplacées par des paramètres de variable de script de génération spécifiques.

Remarque : Lors de l'utilisation du fichier de configuration ant.conf, la variable JAVA_HOME spécifiée remplace toute variable JAVA_BIN spécifiée dans \$GLOBAL pour les compilations de projet Java/J2EE. Ce n'est pas le cas si la configuration Ant est effectuée dans le fichier rsed.envvars,

comme indiqué dans le document *Rational Developer for System z - Guide de configuration de l'hôte* (SC11-6285).

Un exemple de membre BWBLOB est fourni dans la bibliothèque FEK.SFEKSAMV. Cet exemple de membre doit être copié dans le type SCLM J2EEBLD de la hiérarchie SCLM comme membre \$GLOBAL et sauvegardé avec un langage sans analyse valide, tel que TEXT (comme spécifié dans le traducteur de langage FLM@TEXT de la bibliothèque SISPMACS).

Le membre \$GLOBAL fournit les informations suivantes aux procédures de génération JAVA/J2EE :

Tableau 5. Variables \$GLOBAL

Variable	Description
ANT_BIN	Chemin d'accès du répertoire du système de fichiers z/OS UNIX System Services de l'environnement d'exécution Ant Exemple : <property name="ANT_BIN" value="/usr/lpp/apache-Ant-1.6.0/bin/Ant"/>
JAVA_BIN	Chemin d'accès du répertoire du système de fichiers z/OS UNIX System Services de l'environnement de compilation/exécution Java Exemple : <property name="JAVA_BIN" value="/usr/lpp/java/5.0/bin"/>
_SCLMDT_WORK_HOME	Emplacement du répertoire de zone de travail SCLM Developer Toolkit Exemple : <property name="_SCLMDT_WORK_HOME" value="/var/rdz"/>
_SCLMDT_CONF_HOME	Emplacement du répertoire de configuration de SCLM Developer Toolkit Exemple : <property name="_SCLMDT_CONF_HOME" value="/etc/rdz/scldmt"/>
CLASSPATH_JARS	Répertoire de chemin de classes du système de fichiers z/OS UNIX System Services utilisé pour les compilations JAVA. Tous les fichiers jar situés dans ce répertoire seront utilisés dans le chemin de classe. Exemple : <property name="CLASSPATH_JARS" value="/var/rdz/CLASSPATH"/>
TRANTABLE	Fichier VSAM contenant les conversions de nom long/abrégé Exemple : <property name="TRANTABLE" value="FEK.#CUST.LSTRANS.FILE"/>
DEBUG_MODE	Spécifiez "on" si vous souhaitez que Developer Toolkit ne supprime pas les fichiers de génération Java/J2EE du système de fichiers z/OS UNIX System Services. Cela est utile si vous souhaitez que la structure de la génération apparaisse dans le système de fichiers USS à des fins de débogage.

Si ces variables doivent être définies pour tous les niveaux du groupe dans le projet SCLM, il peut s'avérer judicieux de créer un membre \$GLOBAL unique au niveau le plus élevé de la hiérarchie. Lorsque le traducteur de génération

JAVA/J2EE s'exécute, il parcourt la hiérarchie à partir du niveau effectuant la génération et utilise le premier membre \$GLOBAL qu'il trouve dans le type J2EEBLD.

Remarque : Le membre \$GLOBAL doit être enregistré en tant que membre SCLM valide pour que la recherche puisse être effectuée dans la hiérarchie.

Si différents paramètres sont requis, par exemple, au niveau de différents groupes de développement, un membre \$GLOBAL peut être créé dans chacun des groupes de développement.

TRANSLATE.conf

Le fichier TRANSLATE.conf fournit des mots clés permettant de déterminer le mode d'enregistrement du code dans SCLM. Le fichier de configuration contient des mots clés qui déterminent le mode de transfert des fichiers vers le système hôte, selon leur définition de langage. Des mots clés spécifiques déterminent si les fichiers dans un certain type de langage sont binaires, transférés et enregistrés, ou si le source de type texte reste au format ASCII et n'est pas converti par défaut d'ASCII en EBCDIC.

De plus, les définitions de langage SCLM déterminent si les noms de fichiers longs sont convertis en noms d'hôte abrégés valides pour l'enregistrement dans SCLM. Ce mappage de noms longs sur des noms abrégés est contrôlé par la fonction VSAM de conversion de nom long/abrégé SCLM.

TRANSLATE.conf se trouve dans /etc/rdz/sclmdt/CONFIG. Vous pouvez modifier le fichier à l'aide de la commande TSO OEDIT.

L'exemple ci-après illustre le code TRANSLATE.conf, qui doit être personnalisé pour correspondre à votre environnement système. Les lignes mises en commentaire commencent par un astérisque (*).

```

*=====
* cross system sharing
TRANVRLS = NO
*=====
* codepage
CODEPAGE ASCII = IS08859-1
CODEPAGE EBCDIC = IBM-1047
*=====
* ascii/ebcdic translation
TRANLANG JAVABIN
TRANLANG J2EEBIN
TRANLANG J2EEOBJ
TRANLANG TEXTBIN
TRANLANG BINARY
TRANLANG DOC
TRANLANG XLS
*=====
* long/short name translation
LOGLANG JAVA
LOGLANG SQLJ
LOGLANG J2EEPART
LOGLANG JAVABIN
LOGLANG J2EEBIN
LOGLANG J2EEOBJ
LOGLANG DOC
LOGLANG XLS

```

Figure 14. *TRANSLATE.conf* - Fichier de configuration de la conversion ASCII/EBCDIC SCLMDT

Les mots clés suivants sont valides dans le fichier *TRANSLATE.conf* :

Page de codes ASCII

Indique les pages de codes ASCII à utiliser pour la conversion. La valeur par défaut est IS08859-1.

Une directive de page de codes doit exister pour ASCII et pour EBCDIC afin que SCLM Developer Toolkit puisse déterminer comment convertir les fichiers transférés.

Page de codes EBCDIC

Indique les pages de codes EBCDIC à utiliser pour la conversion. La valeur par défaut est IBM-1047.

Une directive de page de codes doit exister pour ASCII et pour EBCDIC afin que SCLM Developer Toolkit puisse déterminer comment convertir les fichiers transférés.

TRANVRLS

Indique si le fichier VSAM de conversion de nom long/abrégé peut être partagé par des systèmes. La valeur par défaut est NO. Les seules valeurs acceptées sont YES et NO.

SCLM utilise la méthode de partage de niveau enregistrement (RLS-Record Level Sharing) VSAM pour permettre le partage du fichier VSAM et conserver l'intégrité dans un environnement partagé. Les fichiers VSAM doivent être définis avec la classe STORAGECLASS appropriée pour l'utilisation de RLS. Pour plus d'informations sur RLS, voir le document *DFSMS Using Data Sets (SC26-7410)*.

TRANLANG

Indique quels types de langage SCLM ne requièrent pas de conversion ASCII/EBCDIC (les fichiers seront transférés en binaire vers le système hôte).

Notez que cette directive ne contient pas de signe égal (=) pour séparer le mot clé TRANLANG et le nom du traducteur de langage (factice).

LONGLANG

Détermine quels types de langages SCLM requièrent la conversion de noms longs en noms abrégés. La conversion de noms longs en noms abrégés implique que le nom de fichier long sur le client (y compris la structure de module de répertoire) soit mappé sur un nom d'hôte valide de 8 caractères et enregistré dans SCLM à l'aide de ce nom d'hôte abrégé converti.

Si le langage SCLM n'est pas indiqué dans le mot clé LONGLANG, le fichier client est considéré comme étant déjà au format de nom d'hôte abrégé (8 caractères ou moins) et est enregistré en tant que tel.

Notez que cette directive ne contient pas de signe égal (=) pour séparer le mot clé LONGLANG et le nom du langage SCLM.

Remarque : Il est possible de remplacer le jeu de valeurs dans le fichier TRANSLATE.conf au niveau d'un SITE et d'un projet SCLM, comme décrit dans «Options spécifiques au projet et SITE».

Options spécifiques au projet et SITE

Un utilitaire a été fourni pour permettre la création de certains paramètres à un niveau d'installation SITE ou au niveau d'un projet SCLM spécifique. Les options pouvant être actuellement configurées sont les suivantes :

- Entrée de code de modification obligatoire
- Désactivation des générations et promotions en avant-plan
- Spécification du système de validation des packages. Actuellement, IBM Breeze for SCLM est le seul système de validation pris en charge.
- Définition des cartes de travail de génération par lots, de promotion et de migration
- Remplacement des paramètres du fichier de configuration TRANSLATE.conf
- Restriction des filtres de liste de projets
- Définition des paramètres par défaut des langues bidirectionnelles (bidi)

Vous pouvez définir toutes ces options ou aucune d'elles. Si elles ne sont pas définies, leur valeur par défaut est utilisée dans les programmes. Certaines de ces options peuvent être définies dans le fichier SITE.conf tandis que d'autres peuvent l'être au niveau d'un certain projet SCLM. Il se peut également qu'il n'existe aucun fichier SITE spécifique, auquel cas, les options ne peuvent être définies qu'au niveau d'un projet SCLM. Vous pouvez remplacer les informations des cartes de travail par vos propres cartes de travail entrées via l'interface IDE.

Cet outil est activé en créant le fichier SITE.conf dans le répertoire z/OS UNIX /etc/rdz/scldmt/CONFIG/PROJECT/ (où /etc/rdz/scldmt est le répertoire etc par défaut de SCLM Developer Toolkit). Ce répertoire est créé lors de la personnalisation de Developer for System z.

Un exemple de fichier SITE.conf est fourni dans le répertoire /usr/lpp/rdz/samples/. Copiez ce répertoire et les instructions en fonction de vos besoins. Vous pouvez modifier le fichier à l'aide de la commande TSO OEDIT. L'exemple ci-après illustre le fichier de configuration SITE.conf.

```

* SCLM Developer Toolkit Site Specific option
* SCM Approver processing applies to this project?
BUILDAPPROVER=NONE
PROMOTEAPPROVER=NONE
*
* Change Code entry on check-in is mandatory?
CCODE=N
*
*
* To allow promotion by architecture definition only,
* set the value of PROMOTEONLYFROMARCHDEF to Y
PROMOTEONLYFROMARCHDEF=N
*
* Foreground or On-line builds/promotes allowed for this project?
FOREGROUNDBUILD=Y
FOREGROUNDPROMOTE=Y
*
* Batch Build default jobcard
BATCHBUILD1=//SCLMBILD JOB (#ACCT),'SCLM BUILD',CLASS=A,MSGCLASS=X,
BATCHBUILD2=//          NOTIFY=&SYSUID,REGION=512M
BATCHBUILD3=//*
BATCHBUILD4=//*
*
* Batch Promote default jobcard
BATCHPROMOTE1=//SCLMPROM JOB (#ACCT),'SCLM PROMOTE',CLASS=A,MSGCLASS=X,
BATCHPROMOTE2=//          NOTIFY=&SYSUID,REGION=128M
BATCHPROMOTE3=//*
BATCHPROMOTE4=//*
*
* Batch Migrate default jobcard
BATCHMIGRATE1=//SCLMMIGR JOB (#ACCT),'SCLM MIGRATE',CLASS=A,MSGCLASS=X,
BATCHMIGRATE2=//          NOTIFY=&SYSUID,REGION=128M
BATCHMIGRATE3=//*
BATCHMIGRATE4=//*
*
* Batch Deployment default jobcard
BATCHDEPLOY1=//SCLMDPLY JOB (#ACCT),'SCLM DEPLOY',CLASS=A,MSGCLASS=X,
BATCHDEPLOY2=//          NOTIFY=&SYSUID,REGION=128M
BATCHDEPLOY3=//*
BATCHDEPLOY4=//*
*
* BUILD Security flag for SAF/RACF security call and possible Surrogate
* ID switch
BUILDSECURITY=N
*
* Project list flag if set to N will stop users selecting * as project
* filter. This may avoid long catalog searches for all SCLM projects.
*
PROJECTLISTALL=Y

```

Figure 15. Exemple de paramètre de projet SCLM spécifique au SITE

Des paramètres de configuration peuvent également être spécifiques au projet pour configurer un projet SCLM. Ces paramètres remplacent les valeurs spécifiques au SITE s'il existe un fichier SITE.conf. Si vous souhaitez définir des valeurs spécifiques au projet, vous devez créer un fichier <project.conf> dans le répertoire /PROJECT, <project> correspondant au nom du projet SCLM (Maj./min. non différenciées).

Un exemple de fichier de configuration de projet est fourni dans le répertoire /usr/lpp/rdz/samples/ (SCLMproject.conf). Copiez cet exemple dans le répertoire PROJECT en utilisant le nom de cible approprié et personnalisez les instructions conformément à vos besoins.

Vous pouvez modifier le fichier à l'aide de la commande TSO OEDIT. L'exemple ci-après illustre le code de configuration du projet.

```
* SCLM Developer Toolkit Project Specific option
*
* SCM Approver processing applies to this project?
BUILDAPPROVER=BREEZE
PROMOTEAPPROVER=BREEZE
*
* Change Code entry on check-in is mandatory?
CCODE=Y
*
* Foreground or On-line builds/promotes allowed for this project?
FOREGROUNDBUILD=N
FOREGROUNDPROMOTE=N
*
* Batch Build default jobcard
BATCHBUILD1=//SCLMBILD JOB (#ACCT),'SCLM BUILD',CLASS=A,MSGCLASS=X,
BATCHBUILD2=//          NOTIFY=&SYSUID,REGION=512M
BATCHBUILD3=//*
BATCHBUILD4=//*
*
* Batch Promote default jobcard
BATCHPROMOTE1=//SCLMPROM JOB (#ACCT),'SCLM PROMOTE',CLASS=A,MSGCLASS=X,
BATCHPROMOTE2=//          NOTIFY=&SYSUID,REGION=128M
BATCHPROMOTE3=//*
BATCHPROMOTE4=//*
*
* BUILD Security flag for SAF/RACF security call and possible Surrogate
* ID switch
BUILDSECURITY=N
* PROMOTE Security flag for SAF/RACF security call and possible
* Surrogate ID switch
PROMOTESECURITY=N
* J2EE DEPLOY security flag for SAF/RACF security call and possible
* Surrogate ID switch
DEPLOYSECURITY=N
```

Figure 16. Exemple de paramètre de projet SCLM spécifique au PROJET

Toutes les options répertoriées sont facultatives. Les valeurs par défaut sont utilisées si rien n'est spécifié dans le fichier SITE.conf ou project.conf.

Tableau 6. Options SITE/PROJET

BUILDAPPROVER=produit de validation/NONE	Spécifiez le nom du produit de validation utilisé pour la procédure de génération. Actuellement, le seul produit pris en charge est IBM Breeze for SCLM, qui est sélectionné à l'aide du mot clé BREEZE. La valeur par défaut est NONE.
PROMOTEAPPROVER=produit de validation/NONE	Spécifiez le nom du produit de validation utilisé pour la procédure de promotion. Actuellement, le seul produit pris en charge est IBM Breeze for SCLM. Si PROMOTEAPPROVER a la valeur BREEZE, les zones spécifiques à Breeze sont affichées lors d'une promotion. La valeur par défaut est NONE.
CCODE=N/Y	Spécifiez Y pour que l'entrée du code de modification soit obligatoire lors d'une restitution. La valeur par défaut est N ; l'entrée du code de modification n'est pas obligatoire.
FOREGROUNDBUILD=Y/N	Spécifiez N pour restreindre les générations en avant-plan. La valeur par défaut est Y ; les générations en avant-plan sont autorisées.
FOREGROUNDPROMOTE=Y/N	Spécifiez N pour restreindre les promotions en avant-plan. La valeur par défaut est Y ; les promotions en avant-plan sont autorisées.

Tableau 6. Options SITE/PROJET (suite)

BATCHBUILD1=Carte de travail 1 BATCHBUILD2=Carte de travail 2 BATCHBUILD3=Carte de travail 3 BATCHBUILD4=Carte de travail 4	Définissez une carte de travail par lots par défaut pour la procédure de génération. Des projets différents peuvent utiliser des classes de travaux ou des codes de compte différents pour que l'option de spécification de cartes de travaux spécifiques à un projet permette ce scénario.
BATCHPROMOTE1=Carte de travail 1 BATCHPROMOTE2=Carte de travail 2 BATCHPROMOTE3=Carte de travail 3 BATCHPROMOTE4=Carte de travail 4	Définissez un travail par lots par défaut pour la procédure de promotion. Des projets différents peuvent utiliser des classes de travaux ou des codes de compte différents pour que l'option de spécification de cartes de travaux spécifiques à un projet permette ce scénario.
BATCHMIGRATE1=Carte de travail 1 BATCHMIGRATE2=Job card 2 BATCHMIGRATE3=Job card 3 BATCHMIGRATE4=Job card 4	Définissez un travail par lots par défaut pour la procédure de migration. Des projets différents peuvent utiliser des classes de travaux ou des codes de compte différents pour que l'option de spécification de cartes de travaux spécifiques à un projet permette ce scénario.
BUILDSECURITY=Y/N	Spécifiez Y afin d'appeler l'appel de sécurité SAF/RACF pour l'étape de génération et effectuez éventuellement une commutation d'ID de substitution. Pour plus d'informations, voir Chapitre 4, «Sécurité SCLM», à la page 47.
PROMOTSECURITY=Y/N	Spécifiez Y afin d'appeler l'appel de sécurité SAF/RACF pour l'étape de promotion et effectuez éventuellement une commutation d'ID de substitution. Pour plus d'informations, voir Chapitre 4, «Sécurité SCLM», à la page 47.
DEPLOYSECURITY=Y/N	Spécifiez Y afin d'appeler l'appel de sécurité SAF/RACF pour l'étape de déploiement et effectuez éventuellement une commutation d'ID de substitution. Pour plus d'informations, voir Chapitre 4, «Sécurité SCLM», à la page 47.
ASCII=page de codes ASCII	Spécifiez la page de codes ASCII devant remplacer celle qui est spécifiée dans TRANSLATE.conf. Par exemple : ASCII=UTF-8
EBCDIC=page de codes EBCDIC	Spécifiez la page de codes EBCDIC devant remplacer celle qui est spécifiée dans TRANSLATE.conf. Par exemple : EBCDIC=IBM-420
TRANLANG=Langage SCLM	Spécifiez un paramètre TRANLANG à ajouter à la liste des paramètres TRANLANG spécifiés dans le fichier TRANSLATE.conf. Par exemple : TRANLANG=DOC
NOTRANLANG=Langage SCLM	Utilisez le mot clé NOTRANLANG pour supprimer un mot clé TRANLANG déjà spécifié de la liste des mots clés autorisés pour ce projet SCLM, conformément au fichier TRANSLATE.conf. Par exemple : NOTRANLANG=JAVA
LOGLANG=Langage SCLM	Spécifiez un paramètre LOGLANG à ajouter à la liste des paramètres LOGLANG spécifiés dans le fichier TRANSLATE.conf. Par exemple : LOGLANG=DOC

Tableau 6. Options SITE/PROJET (suite)

NOLONGLANG=Langage SCLM	Utilisez le mot clé NOLONGLANG pour supprimer un mot clé LONGLANG déjà spécifié de la liste des mots clés autorisés pour ce projet SCLM, conformément au fichier TRANSLATE.conf. Par exemple : NOLONGLANG=COBOL
BIDIPROP=LANG=SCLM Language/* TextOrient=LTR/RTL TextType=Visual/Logical SymetricSwap=On/Off NumericSwap=On/Off	Utilisez le mot clé BIDIPROP pour affecter les langages SCLM comme langages bidirectionnels par défaut. Le paramètre LANG= peut correspondre à tous les langages SCLM à des langages SCLM spécifiques. Le support bidirectionnel n'est pris en charge que sous Developer for System z.
PROJECTLISTALL=Y	Si l'option de liste de projets a la valeur N, les utilisateurs ne pourront plus sélectionner * comme filtre de projets. Cela peut éviter les longues recherches dans les catalogues utilisateur pour tous les projets SCLM.

Exemple d'utilisation de combinaisons de remplacements de fichier TRANSLATE.conf

Le fichier TRANSLATE.conf définit les paramètres par défaut de la prise en charge des pages de codes et le support de langage SCLM par défaut à appliquer dans SCLM Developer Toolkit. Dans cet exemple, TRANSLATE.conf possède les valeurs répertoriées ci-après.

```
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
```

Il est possible pour des projets SCLM différents qui stockent des types de données différents, éventuellement dans d'autres langues, de remplacer ces paramètres par défaut. Le fichier de configuration du projet SCLM SCLMPRJ1.conf peut donc contenir les paramètres de substitution suivants :

```
* Arabic Codepage overrides
*
ASCII=UTF-8
EBCDIC=IBM-420
*
* Project specific TRANLANG and LONGLANG entries
*
TRANLANG=DOC
LONGLANG=DOC
```

Cela affecte aux pages de codes des traductions source la paire de pages de codes arabes. De plus, le langage SCLM DOC est ajouté aux valeurs par défaut du fichier TRANSLATE.conf.

Le projet SCLM SCLMPRJ2 peut contenir des paramètres de remplacement différents dans le fichier SCLMPRJ2.conf :

```
* Hebrew Codepage overrides
*
ASCII=UTF-8
EBCDIC=IBM-424
*
* Project specific TRANLANG and LONGLANG entries
*
TRANLANG=DOC
TRANLANG=XLS
NOTRANLANG=JAVABIN
NOTRANLANG=J2EEBIN
NOTRANLANG=J2EEOBJ
LONGLANG=DOC
LONGLANG=XLS
NOLONGLANG=COBOL
NOLONGLANG=J2EEPART
NOLONGLANG=JAVABIN
NOLONGLANG=J2EEBIN
NOLONGLANG=J2EEOBJ
```

Cela affecte aux pages de codes des traductions source la paire de pages de codes hébreux. De plus, les langages SCLM DOC et XLS sont ajoutés aux valeurs par défaut du fichier TRANSLATE.conf. Dans ce cas, toutefois, les valeurs par défaut définies dans le fichier TRANSLATE.conf sont ensuite supprimées. Cela n'est pas vraiment nécessaire car la présence de paramètres supplémentaires ne pose pas de difficultés, mais cela montre comment un projet peut être configuré de sorte à ne contenir que les langages SCLM requis pour un projet SCLM spécifique.

Exemple d'utilisation de combinaisons de remplacements de BIDIPROP

Les valeurs BIDIPROP spécifiées dans le fichier SITE.conf peuvent être remplacées par les valeurs BIDIPROP spécifiées dans les fichiers SCLM <project>.conf spécifiques aux projets. Par exemple, les valeurs suivantes sont définies dans le fichier SITE.conf :

```
*
* ----- SITE SPECIFIC BIDI OPTIONS -----
*
* BiDi Language default properties
*
BIDIPROP=LANG=*      TextOrient=LTR TextType=Visual SymetricSwap=Off NumericSwap=Off
```

Cela affecte à tous les langages SCLM les paramètres spécifiés. Il est maintenant possible de spécifier les valeurs suivantes dans le fichier ADMIN10.conf :

```
*
* BiDi Language default properties
BIDIPROP=LANG=JAVA TextOrient=RTL TextType=Visual SymetricSwap=On NumericSwap=Off
BIDIPROP=LANG=COBOL TextOrient=RTL TextType=Logical SymetricSwap=Off NumericSwap=Off
```

Ces paramètres remplacent ceux du fichier SITE.conf pour les définitions de langage JAVA et COBOL. Tous les autres langages possèdent les paramètres par défaut, comme spécifié dans SITE.conf.

Chapitre 3. Support SQLJ

SQLJ est une extension de langage pour Java. Il s'agit de l'une des technologies qui permettent aux programmeurs Java d'inclure des communications de base de données dans leurs programmes. SQLJ permet de générer du code SQL imbriqué, statique, qui s'avère généralement plus performant que ses équivalents dynamiques, tels que JDBC.

SCLM Developer Toolkit est livré avec des scripts exemple vous permettant de générer des programmes Java activés par SQLJ à l'aide de DB2.

Après la lecture de ce chapitre, vous devriez comprendre le fonctionnement de base de SQLJ et savoir comment appliquer ces connaissances lors de l'utilisation de SCLM Developer Toolkit.

Présentation de SQL

SQL est l'acronyme de *Structured Query Language*. Il s'agit d'un langage ouvert permettant d'interroger, d'ajouter, de supprimer et de modifier des données dans un système RDMS (Relational Database Management System).

La première implémentation de ce langage a été effectuée dans un produit de base de données IBM pionnier dans les années 1970 : System R. Depuis, SQL s'est développé, a été standardisé (par ANSI et ISO) et a pris plusieurs formes sur un grand nombre de systèmes de base de données.

Présentation de DB2

DB2 est un système de base de données populaire, utilisé à l'origine pour la plateforme de l'ordinateur principal, mais qui depuis a été étendu à de nombreuses autres plateformes. Il s'agit de la norme pour les systèmes de gestion de base de données relationnelle sous z/OS.

DB2 UDB Version 8 correspond à la version utilisée par les scripts de génération SCLM Developer Toolkit. Dans le présent chapitre, toute référence à DB2 s'applique spécifiquement à DB2 UDB Version 8.

Qu'est-ce que JDBC ?

JDBC signifie *Java Database Connectivity*. En développement Java, cette technologie est bien connue et couramment utilisée pour implémenter l'interaction des bases de données. JDBC est une API au niveau appel, ce qui signifie que les instructions SQL sont transmises sous forme de chaînes à l'API, qui les exécute ensuite sur le système RDMS. Par conséquent, la valeur de ces chaînes peut être modifiée lors de l'exécution et rendre ainsi JDBC dynamique.

Les programmes JDBC sont exécutés plus lentement que leurs équivalents SQLJ, mais ils bénéficient du concept "Write once, call anywhere". Comme aucune interaction n'est requise lors de la phase d'exécution, un programme JDBC est donc particulièrement transférable et peut passer d'un système à un autre pour un coût minimal.

Qu'est-ce que SQLJ ?

SQLJ est une extension de langage utilisée pour les transactions de base de données dans les applications Java. Il génère un SQLJ imbriqué, statique. Ce terme se compose du sigle SQL (*Structured Query Language*) suivi de la lettre J, pour *Java*.

SQLJ est *statique* car les instructions SQL qui seront exécutées sont connues lors de l'assemblage du programme, contrairement à JDBC où les requêtes qui sont exécutées peuvent être modifiées à tout moment.

SQLJ est *imbriqué* car, lors de la liaison, une forme sérialisée des instructions SQL des programmes est fournie à la base de données. Cette dernière utilise ces données sérialisées pour déterminer les chemins d'accès optimisés aux tables qui y sont référencées. Dans JDBC, la base de données n'a pas moyen de déterminer quelles instructions seront exécutées, jusqu'à ce qu'elle les reçoit de l'application lors de la phase d'exécution. Par conséquent, il doit déterminer les chemins d'accès lors de la phase d'exécution. Le temps système est alors augmenté. Vous pouvez éviter cette situation en utilisant SQLJ.

Comparaison de JDBC et SQLJ

Cette table se base sur les informations de la section 5.2 du Redbook *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*.

Tableau 7. Comparaison de JDBC et SQLJ

	SQLJ (statique)	JDBC (dynamique)
PERFORMANCES	La plupart du temps, le SQL statique est plus rapide que le SQL dynamique car lors de la phase d'exécution, seules les autorisations des packages et des plans doivent être vérifiées avant l'exécution du programme.	Les instructions SQL dynamiques doivent être analysées, les autorisations des tables/vues doivent être vérifiées et le chemin d'optimisation doit être déterminé.
AUTORISATION	Avec SQLJ, le propriétaire de l'application octroie les droits d'exécution (EXECUTE) sur le plan ou le package et le destinataire de ces droits doivent exécuter l'application, tel que c'est spécifié.	Avec JDBC, le propriétaire de l'application octroie des droits sur toutes les tables sous-jacentes utilisées par l'application. Le destinataire de ces droits peut effectuer toute opération autorisée par ces droits. Par exemple, les utiliser en dehors de l'application pour laquelle ces droits ont été octroyés à l'origine. L'application ne peut pas contrôler quelles opérations l'utilisateur peut effectuer.
DEBOGAGE	SQLJ n'est pas une API, mais une extension de langage. Cela signifie que les outils SQLJ ont connaissance des instructions SQL dans votre programme et qu'ils vérifient si leur autorisation et leur syntaxe sont correctes lors de la procédure de développement du programme.	JDBC est une pure API au niveau appel. Autrement dit, le compilateur Java ne prend pas en compte les instructions SQL. Elles apparaissent uniquement en tant qu'arguments d'appels de méthode. Si l'une de vos instructions est erronée, vous ne détecterez pas cette erreur jusqu'à ce que la base de données affiche une erreur.

Tableau 7. Comparaison de JDBC et SQLJ (suite)

	SQLJ (statique)	JDBC (dynamique)
SURVEILLANCE	Avec SQLJ, la surveillance du système et les rapports de performances sont bien meilleurs. Les packages SQL statiques vous indiquent les noms des programmes exécutés à un moment donné. Ces packages sont utiles pour l'étude de l'utilisation de l'unité centrale par les différentes applications, le verrouillage de certains éléments (blocage ou délai, par exemple), etc.	Alors que dans SQLJ, vous pouvez déterminer le nom du programme en cours d'exécution, avec JDBC, toutes les transactions sont effectuées via le même programme. Cela rend la surveillance et l'identification des zones d'incident plus difficiles.
PROLIXITE	Les instructions SQLJ étant codées dans une syntaxe purement SQL, sans qu'il ne soit nécessaire de les encapsuler dans une méthode Java, les programmes sont plus faciles à lire et donc à gérer. En outre, comme une partie du code standard codé explicitement en JDBC est généré automatiquement en SQLJ, les programmes écrits en SQLJ ont tendance à être plus courts que les programmes JDBC équivalents.	Avec JDBC, toutes les instructions SQL doivent être encapsulées dans des appels d'API, ce qui donne généralement un code peu clair et prolixe.

Présentation d'un profil sérialisé

Un profil sérialisé est un code écrit en SQLJ est placé dans un fichier ayant l'extension `.sqlj`. Dans la première étape de préparation de programme (qui sera décrit en détails ultérieurement), le fichier `.sqlj` est transmis au traducteur SQLJ.

Ce traducteur génère deux types de sortie. La première est le code source Java (`.java`). Ce code source correspond à l'implémentation Java du code dans le fichier `.sqlj`.

Le deuxième type de sortie est un profil sérialisé (`.ser`). Ce fichier contient toutes les instructions SQL provenant du fichier `.sqlj` à un format sérialisé. Ce profil doit être disponible lors de l'exécution pour le programme. Il peut également être utilisé pour la liaison au système de gestion de base de données relationnelle.

Présentation d'un module DBRM

DBRM signifie *module d'interrogation de base de données*. Il s'agit de la représentation sérialisée DB2 standard des instructions SQL dans un programme. Par exemple, un programme peut être écrit en COBOL. Ce programme sera prétraité par DB2 pour générer un module DBRM permettant d'accéder à un sous-système DB2 particulier.

Avec SQLJ, la procédure est légèrement différente et est appelée, dans la terminologie DB2 UDB Version 8, *mode de compatibilité*. L'utilitaire `db2sqljcustomize` peut être fourni avec des arguments de ligne de commande facultatifs qui entraînent la génération d'un module DBRM. Ce module DBRM peut ensuite être associé à DB2 à l'aide de moyens traditionnels, tel qu'un script REXX appelé par un exit utilisateur SCLM.

Préparation du programme SQLJ

Avant d'aborder l'utilisation de SCLM Developer Toolkit pour générer des programmes SQLJ, examinons d'abord la procédure manuelle. Cette procédure concerne l'implémentation DB2 de SQLJ et se compose de trois commandes appelées *sqlj*, *db2sqljcustomize* et *db2bind*. Notez que l'étape de liaison pouvant être éventuellement effectuée dans *db2sqljcustomize*, *db2bind* n'est pas toujours nécessaire.

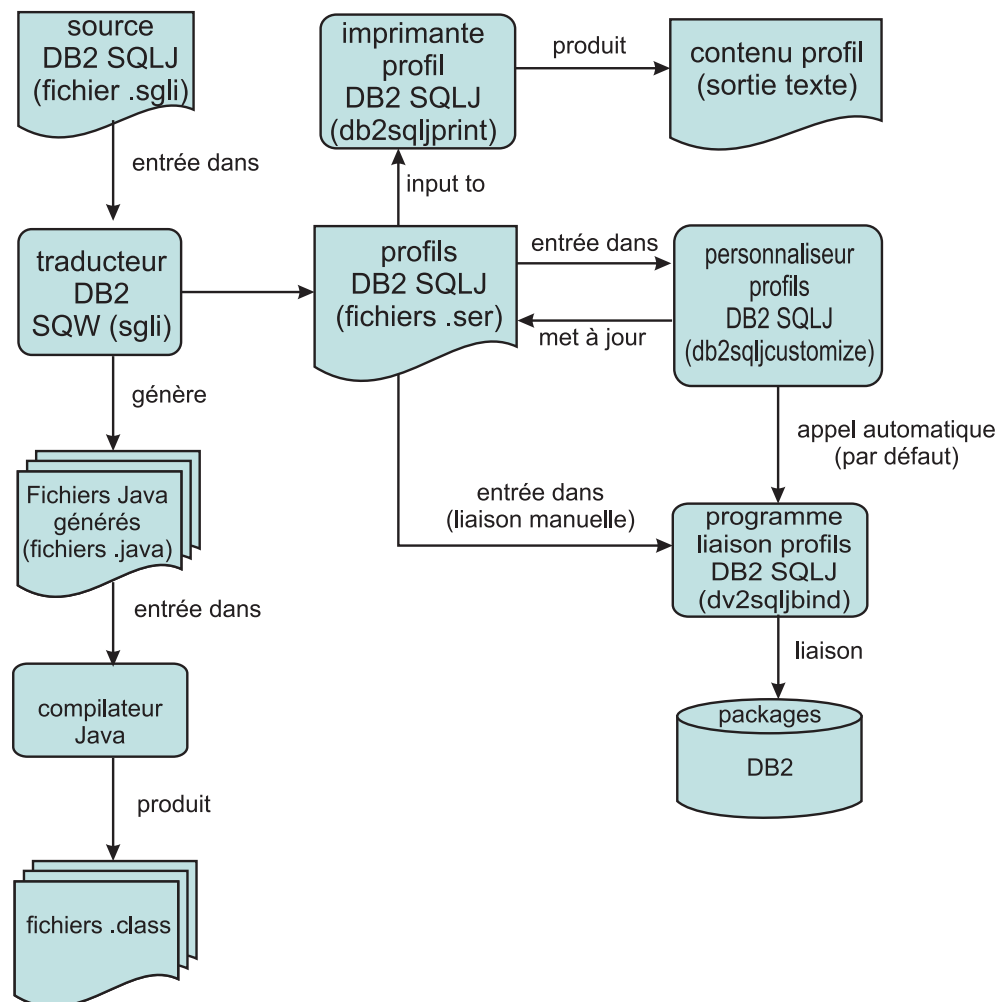


Figure 17. Préparation du programme SQLJ

Traduction

Le traducteur SQLJ (à ne pas confondre avec le *traducteur de langage SCLM*) utilise les fichiers source SQLJ en tant qu'entrée et génère du code source Java (fichiers .java) et des profils sérialisés (fichiers .ser).

Le langage SQLJ lui-même n'est pas abordé dans le présent document. Pour obtenir des références sur le développement de code SQLJ, consultez le site <http://www.sql.org>.

Le nombre de profils sérialisés générés par fichier .sqlj dépend du nombre de classes de *contexte de connexion* référencées dans le code SQLJ. Un profil sérialisé sera généré pour chacune d'entre elles.

De nombreux fichiers source SQLJ ne font référence qu'à une seule classe de contexte de connexion et ne génèrent donc qu'un seul profil sérialisé. Les profils sérialisés sont nommés en fonction de l'ordre dans lequel ils sont référencés dans le fichier source. Le nom se présente au format suivant :

nomprog_SJProfileX.ser

Où :

- *nomprog* représente le nom du programme. Ce nom est déterminé en supprimant l'extension .sqlj du nom de fichier source en entrée.
- *X* représente un entier représentant l'index de la classe en cours. L'index est basé sur la valeur zéro. La première classe de contexte de connexion référencée va générer le profil 0, la deuxième va générer le profil 1, etc.

Exemple :

Entrée : Customer.sqlj (fait référence à une classe de contexte de connexion)

Sortie : Customer.java
Customer_SJProfile0.ser

Et éventuellement, si l'argument `-compile=true` est fourni à `sqlj` :
Customer.class

Personnalisation

Une fois que les profils sérialisés sont générés, nous devons les personnaliser. La commande permettant d'effectuer cette action dans DB2 version 8 est *db2sqljcustomize*. Dans les versions précédentes, la commande était *db2profsc*. Chaque appel du personnaliseur doit correspondre à un appel du traducteur SQLJ. Autrement dit, si un simple appel du traducteur a généré cinq profils, ces cinq profils doivent être transmis en tant qu'entrée à un appel unique du personnaliseur de profil. Il est également possible d'associer chaque nom de programme à un appel de chacun des utilitaires. N'oubliez pas que le nom du programme est identique au nom de fichier source en entrée, sans l'extension .sqlj.

La personnalisation ajoute des informations propres à DB2 au profil sérialisé utilisé lors de l'exécution. D'autres options, telles la liaison automatique, peuvent être configurées à l'aide d'options de ligne de commande. Si vous utilisez une version existante de DB2 ou que vous spécifiez les options *gendbrm* et *dbrmdir* pour *db2sqljcustomize*, un fichier de module DBRM est généré. Ce fichier est utilisé par la suite pour accéder à la base de données. A l'aide du pilote universel de DB2 UDB 8+, vous pouvez ignorer la génération d'un module d'interrogation de base de données à l'aide des profils sérialisés générés par le traducteur SQLJ.

Liaison

La liaison est la dernière étape de la procédure de préparation du programme SQLJ. Dans DB2 version 8, la commande de liaison est *db2sqljbind* ou, pour une liaison automatique, *db2sqljcustomize*. La liaison constitue l'étape qui génère un chemin d'accès aux tables DB2 pour vos instructions SQL sérialisées. Ces instructions sont disponibles sous forme de module DBRM ou de profil sérialisé.

Par défaut, quatre packages sont créés ; un pour chaque niveau d'isolement. Vous pouvez effectuer la liaison à l'aide de la méthode traditionnelle lorsqu'un module d'interrogation est utilisé ou de la nouvelle méthode Universal lorsque les profils sérialisés sont utilisés à la place.

Traducteurs et types SCLM DT

Avant d'aborder les traducteurs et les types SCLM, il est important de faire la distinction entre un *traducteur de langage SCLM* ou, plus simplement, un *traducteur SCLM* et le traducteur SQLJ *sqlj* fourni avec DB2.

Dans SCLM, tout langage défini doit posséder un traducteur pour que le système sache comment traiter ce langage. Ce traducteur est différent du traducteur SQLJ, *sqlj*, un utilitaire de ligne de commande qui génère des profils sérialisés et un code source Java à partir d'un fichier source SQLJ.

Une fois cette distinction établie, nous étudierons les *types SCLM* et les *traducteurs SCLM* associés à la procédure de génération SQLJ, de la manière suivante :

Un traducteur SCLM pour SQLJ est fourni et doit être affecté comme type de langage de tout le code source SQLJ stocké dans SCLM. Ce nouveau traducteur requiert la définition de types SCLM supplémentaires. Le traducteur SCLM de SQLJ est similaire au traducteur JAVA, mais il contient des définitions IOTYPE supplémentaires pour les types de sortie SCLM SQLJSER et DBRMLIB. Si vous ne souhaitez pas générer de fichier de module d'interrogation de base de données comme partie de l'étape de personnalisation, cet élément DBRMLIB IOTYPE peut être retiré de la définition de langage SQLJ.

Dans la définition de projet, un administrateur doit définir et générer le traducteur SCLM et les types supplémentaires.

Tableau 8. Types de traducteur SCLM pour SQLJ

SQLJSER	Requis pour stocker les fichiers de profil sérialisé générés (fichiers .ser) créés ou personnalisés dans les étapes de traduction et de personnalisation. Il est recommandé de définir ce fichier de type SCLM au format recfm=VB, lrecl=256.
DBRMLIB	Type devant contenir les fichiers DBRM générés créés dans l'étape de personnalisation. Ce type est requis uniquement pour les clients utilisant les fichiers de modules d'interrogation de base de données générés dans le cadre du traitement de liaison DB2.

Personnalisation de la procédure de génération

Pour conserver une flexibilité maximale, la procédure de génération SQLJ peut être très personnalisée, pour répondre aux besoins de différentes configurations de site et de toute combinaison de paramètres à transmettre à *sqlj* et *db2sqljcustomize*.

Cette section décrit les concepts liés à l'implémentation SQLJ de SCLM Developer Toolkit. Après la lecture de cette section, vous devriez pouvoir personnaliser le processus de génération afin qu'il réponde aux exigences de votre site.

Lors de la traduction SQLJ et de la personnalisation des profils, SCLM Developer Toolkit appelle directement les mêmes classes Java utilisées par les commandes *sqlj* et *db2sqljcustomize*. Sachez que les arguments fournis aux procédures de traduction et de personnalisation SCLM DT sont identiques. Pour obtenir une description

détaillée de tous les arguments de ligne de commande pour chaque commande, consultez le document *DB2 Universal Database User Guide*.

Personnalisation du script de génération

En supposant que vous ayez utilisé l'assistant Ajouter à SCLM, le script de génération de votre programme SQLJ reçoit le même nom de membre que l'Archdef. Par exemple, si la définition d'archive de votre projet sqlj est SCLM10.DEV1.ARCHDEF(SQLJ01), vous trouverez le script de génération dans SCLM10.DEV1.J2EEBLD(SQLJ01).

Ce script de génération contient une référence au script de génération maître. Cet élément est disponible dans la propriété. La configuration répertoriée pour les étapes de traduction et de personnalisation se poursuit dans ce fichier.

Remarque : Le script de génération livré avec Developer Toolkit est BWBSQLB pour les projets JAR et BWBSQLBE pour les projets EJB. Il n'est pas nécessaire de changer cette valeur.

sqlj.* properties

Chaque propriété répertoriée dans le tableau 9 apparaît dans le script de génération BWBSQLB. Ces propriétés sont au format XML, comme cela est présenté ci-dessous :

La configuration du script implique la modification de la valeur pour toute propriété appropriée.

Tableau 9. sqlj.* properties

Nom	Valeur	Description
sqlj.exec	usr/lpp/rdz/bin/bwbsqlc.rex	Spécifie l'emplacement de la routine d'exécution sqlj & db2sqljcustomize bwbsqlc.rex, qui se trouve dans le répertoire d'installation de Developer for System z.
sqlj.class	sqlj.tools.Sqlj	Spécifie le nom de la classe sqlj. Il s'agit du nom de la classe appelée par l'utilitaire sqlj. Il est peut probable que vous deviez modifier cette valeur.
sqlj.bin	/db2path/bin	Spécifiez le répertoire db2 sqlj bin dans lequel se trouve le script sqlj.
sqlj.cp	/db2path/jcc/classes/sqlj.zip	Spécifie l'emplacement du fichier sqlj.zip à inclure dans le chemin d'accès aux classes.
sqlj.arg	-compile=false status linemap=NO db2optimize	Spécifie les arguments de propriété globaux ci-après pour le traitement sqlj.

db2sqljcustomize.* properties

Chaque propriété répertoriée dans le tableau 10, à la page 42 apparaît dans le script de génération BWBSQLB. Ces propriétés sont au format XML, comme cela est présenté ci-dessous :

```
<property name= NOM value= VALEUR />
```

La configuration du script implique la modification de la valeur pour toute propriété appropriée.

Tableau 10. *db2sqljcustomize.* properties*

Nom	Valeur	Description
sqljdb2cust.class	com.ibm.db2.jcc.sqlj.Customizer	Spécifie le nom de la classe de personnalisation sqlj db2. Il est peut probable que vous deviez modifier cette valeur.
db2sqljcust.cp	/db2path/jcc/classes/db2jcc.jar: ./SRC: /db2path/jcc/classes/db2jcc_license_cisuz.jar	Paramètres du chemin d'accès aux classes pour l'utilitaire de personnalisation. Les noms de chemin complets doivent être fournis en XML.
db2sqljcust.arg	-automaticbind NO -onlinecheck YES -staticpositioned YES -bindoptions â ISOLATION(CS)â -genDBRM	Arguments généraux à fournir à l'utilitaire de personnalisation.
db2sqljcust.propfile	user.properties	Nom du fichier de propriétés temporaire à transmettre à un script de détermination des propriétés utilisateur pour les valeurs de propriétés dynamiques. Conservez la valeur par défaut.
db2sqljcust.userpgm	AUCUN si vous souhaitez ignorer le script. Sinon, spécifiez le nom de fichier et le chemin d'accès complet du script utilisateur.	Ce script est exécuté immédiatement avant l'utilitaire de personnalisation. Il met à jour de manière dynamique un fichier de propriétés utilisé en entrée dans l'utilitaire de personnalisation.

Script utilisateur personnalisé

Le script de génération SQLJ fourni par SCLM Developer Toolkit est conçu pour fonctionner en mode de compatibilité DB2 UDB v8. Ce mode prend en charge le concept DB2 des modules d'interrogation de base de données et non la liaison via les profils sérialisés. Pour utiliser les profils sérialisés, vous devez modifier le script BWBSQLB. Cette opération est abordée dans la sous-rubrique «Liaison [Profil sérialisé]», à la page 45.

La méthodologie de liaison à part, pour faire correspondre la procédure de génération à votre site, vous devrez probablement personnaliser les arguments de *sqlj* et *db2sqljcustomize* en fonction de votre environnement de base de données, de vos règles d'isolement et d'autres facteurs. Vous pouvez également transmettre vos propres scripts afin de déterminer les propriétés dynamiques pour ces arguments. Par exemple, vous pouvez vouloir créer un nom de package lié au nom du fichier d'entrée.

SCLM Developer Toolkit vous le permet ; vous n'avez qu'à spécifier votre propre script de personnalisation. L'intégralité de la procédure de création XML ANT repose sur le concept des "propriétés", les éléments XML *Property* spécifiant une paire de noms ou de valeurs. Par exemple, dans l'étape *db2sqljcustomize* du script de génération BWBSQLB, les arguments de ligne de commande globaux à fournir à *db2sqljcustomize* sont définis dans un élément de propriété appelé *db2sqljcust.arg* et dont la valeur par défaut est *-automaticbind NO -onlinecheck YES -staticpositioned YES -bindoptions "ISOLATION(CS)" genDBRM lang=EN-AU*.

Si vous souhaitez modifier les arguments fournis, vous pouvez éditer le script de génération fourni pour modifier la valeur de la propriété, ce qui modifie les paramètres de manière globale, ou ancrer votre propre script de personnalisation dans la procédure.

Pour intégrer votre propre script de propriété personnalisé, indiquez le nom du script dans `db2sqljcust.userpgm` et le nom du fichier de propriétés à placer dans `db2sqljcust.propfile`.

Le script spécifié dans `db2sqljcust.userpgm` est exécuté immédiatement avant le processus `db2sqljcustomize`. Votre script met à jour dynamiquement le fichier de propriétés spécifié dans `db2sqljcust.userpgm`. Ce fichier de propriétés est utilisé en entrée du processus `db2sqljcustomize` car la procédure de génération concatène les deux propriétés dans le fichier de propriétés mis à jour de manière dynamique et les propriétés déjà définies dans le script de génération.

Le script spécifié dans `db2sqljcust.userpgm` sera fourni par les arguments suivants lors de l'exécution :

Argument	Description
Basedir	Répertoire de base (répertoire de l'espace de travail)
Propfile	Nom du fichier de propriétés à créer et mettre à jour. Remarque : Le fichier de propriétés doit être <code>basedir'/propfile</code> .
Sqljf	Liste de noms de fichier, représentant les profils sérialisés (.ser) à traiter par <code>db2sqljcustomize</code> .

Les propriétés doivent être définies dans le fichier au format suivant, avec une déclaration de propriété par ligne :

```
argument=value
ex.
singlepkgname= pkgname
```

Exemple :

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

La routine de personnalisation est appelée une fois par fichier. Enfin, les propriétés des arguments sont utilisées pour générer la chaîne d'arguments requise pour l'appel `db2sqljcustomize`. Exemple :

```
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}
```

Liaison [DBRM]

DB2 utilise généralement un a *module d'interrogation de base de données* (ou module DBRM) pour la liaison. Le module DBRM est généré par la commande `db2sqljcustomize` lorsque l'option `gendbrm` est fournie. Sans cette option, la commande suppose que vous souhaitez effectuer la liaison via des profils sérialisés et ne génère pas de module DBRM.

Si vous indiquez ce paramètre, SCLM Developer Toolkit utilise les modules d'interrogation de base de données et les enregistre dans SCLM pour une

utilisation future. Cette technique présente l'avantage de pouvoir facilement effectuer une liaison DB2 dans un exit utilisateur SCLM, tel que l'exit de génération/copie.

Etant donné qu'une liste d'objets mis à jour est automatiquement fournie pour un exit de génération/copie, vous pouvez lier à nouveau de manière sélective uniquement les modules qui ont été modifiés, ce qui évite l'inefficacité via la liaison redondante.

Quatre étapes sont requises pour la configuration de la liaison des systèmes de gestion de bases de données relationnelle :

1. Définissez les arguments appropriés pour *sqlj*, de la manière suivante :

```
<!-- specify global property arguments below for sqlj processing -->
<property name="sqlj.arg"
  value="-compile=false -status -linemap=no"/>
```

Argument	Description
compile=false	Si cette option a la valeur false, le traducteur sqlj ne peut pas compiler automatiquement la source Java qu'il génère. SCLM Developer Toolkit utilisant le source généré dans sa propre procédure de génération, il est recommandé de toujours spécifier la valeur false.
linemap=no	Spécifie si les numéros de ligne des exceptions Java correspondent aux numéros de ligne des fichiers source SQLJ (fichier .sqlj) ou du fichier source Java généré par les fichiers du traducteur SQLJ (fichier .java). Un fichier .class étant pour cela requis, cette option doit avoir la valeur <i>no</i> lorsqu'elle est utilisée avec <i>compile=false</i> .
status	Imprime l'écran d'état immédiat du traitement SQLJ.

2. Définissez les arguments appropriés pour *db2sqljcustomize*, y compris *gendbrm*, de la manière suivante :

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

Argument	Description
automaticbind no	Si cette option a la valeur no, le personnalisateur n'effectue pas de liaison une fois la personnalisation terminée.
onlinecheck yes	Effectue une vérification en ligne sur le système spécifié par le paramètre <i>url</i> . La valeur par défaut est yes si <i>url</i> est fourni et no dans le cas contraire.
Bindoptions ISOLATION(CS)	Demande au programme de liaison de créer un seul package (lecture non reproductible). Option utilisée avec <i>singlepkgname</i> (défini de manière dynamique).
gendbrm	Demande au personnalisateur de générer des fichiers DBRM.

3. Configurez le script utilisateur.

Définissez l'emplacement de votre programme utilisateur dans BWBSQLB. Indiquez au processus de génération où trouver le script rex permettant de calculer les propriétés dynamiques.

La propriété importante que nous souhaitons configurer de manière dynamique est *singlepkgname*. Il s'agit du nom du package cible de la liaison. Chaque programme possède son propre nom de package unique qui, dans cet exemple simple, correspond aux huit premières lettres du nom du programme.

4. Créez un exit de génération pour lier le module DBRM. L'exit de génération/copie est recommandé.

Comme nous utilisons *singlepkgname* dans l'étape de personnalisation, le nom du package est identique au nom du module DBRM.

Liaison [Profil sérialisé]

La nouvelle approche recommandée pour la liaison des programmes SQLJ consiste à utiliser les profils sérialisés (fichiers `.ser`) pour effectuer la liaison. Cela était inévitable car le profil sérialisé effectue la même fonction que le module DBRM : fournir une image sérialisée des instructions du programme.

En modifiant légèrement le script de génération BWBSQLB, vous pouvez configurer SCLM Developer Toolkit pour qu'il utilise cette méthode à la place. Il s'agit simplement d'une question de modification des arguments fournis à `db2sqljcustomize` pour supprimer le commutateur de ligne de commande `gendbrm` et affecter à `automaticbind` la valeur YES.

Trois étapes sont requises pour la configuration de la liaison des profils sérialisés :

1. Définissez les arguments appropriés pour `sqlj`, de la manière suivante :

Il n'existe pas d'arguments de ligne de commande pour le traducteur `sqlj` qui sont réservés à la liaison des profils sérialisés. Toutefois, les arguments définis pour cet exemple particulier sont illustrés.

```
<!-- specify global property arguments below for sqlj processing -->
<property name="sqlj.arg"
value="-compile=false -status -linemap=no"/>
```

Argument	Description
<code>compile=false</code>	Si cette option a la valeur <code>false</code> , le traducteur <code>sqlj</code> ne peut pas compiler automatiquement la source Java qu'il génère. SCLM Developer Toolkit utilisant le source généré dans sa propre procédure de génération, il est recommandé de toujours spécifier la valeur <code>false</code> .
<code>linemap=no</code>	Spécifie si les numéros de ligne des exceptions Java correspondent aux numéros de ligne des fichiers source SQLJ (fichier <code>.sqlj</code>) ou du fichier source Java généré par les fichiers du traducteur SQLJ (fichier <code>.java</code>). Un fichier <code>.class</code> étant pour cela requis, cette option doit avoir la valeur <code>no</code> lorsqu'elle est utilisée avec <code>compile=false</code> .
<code>status</code>	Imprime l'écran d'état immédiat du traitement SQLJ.

2. Définissez les arguments appropriés pour `db2sqljcustomize`, de la manière suivante :

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

Argument	Description
<code>automaticbind yes</code>	Si cette option a la valeur <code>yes</code> , le personnaliseur effectue également la liaison une fois la personnalisation terminée. Si elle a la valeur <code>no</code> , la liaison doit être effectuée de manière distincte, à l'aide de la commande <code>db2bind</code> .
<code>onlinecheck yes</code>	Effectue une vérification en ligne sur le système spécifié par le paramètre <code>url</code> . La valeur par défaut est <code>yes</code> si <code>url</code> est fourni et <code>no</code> dans le cas contraire.

3. Configuration du script d'utilisateur.

Définissez l'emplacement de votre programme utilisateur dans BWBSQLB. Indique au processus de génération où trouver le script `rex` permettant de calculer les propriétés dynamiques.

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

Chapitre 4. Sécurité SCLM

SCLM Developer Toolkit offre des fonctionnalités de sécurité facultatives pour les fonctions de génération, de promotion et de déploiement.

- La fonction de sécurité est contrôlée par les options de sécurité définies dans les fichiers de configuration de SCLM Developer Toolkit et par les profils de votre produit de sécurité.
- Si l'option de sécurité est définie pour une fonction et que l'ID utilisateur demandeur satisfait la vérification des droits par votre produit de sécurité, la fonction peut continuer. Si la vérification des droits du demandeur échoue, la fonction se termine avec le code retour RC=8 et un message d'erreur de sécurité.
- Si elle est définie de cette manière dans votre produit de sécurité, la fonction continue à l'aide d'un autre ID utilisateur (substitut).

Option de sécurité

Vous pouvez définir une option de sécurité de génération, de promotion ou de déploiement dans les fichiers de configuration du SITE/PROJET. Pour en savoir plus sur les fichiers de configuration du SITE/PROJET, voir «Options spécifiques au projet et SITE», à la page 28. Les instructions suivantes contrôlent l'option de sécurité des fonctions de génération, promotion et déploiement, respectivement :

- BUILDSECURITY = Y
- PROMOTSECURITY = Y
- DEPLOYSECURITY = Y

Configuration dans votre produit de sécurité

La sécurité des fonctions de génération, de promotion et de déploiement SCLM utilise l'interface de sécurité SAF/RACROUTE, qui est prise en charge par tous les principaux produits de sécurité.

Les exemples de commande répertoriés concernent RACF. Si vous utilisez un autre produit, reportez-vous à la documentation de votre produit de sécurité.

- A l'aide de la commande RDEFINE, définissez dans RACF toutes les ressources appartenant aux classes spécifiées dans la table des descripteurs de classe. La commande RDEFINE ajoute un profil pour la ressource dans la base de données RACF pour contrôler l'accès à cette ressource.
- La commande PERMIT permet de gérer les listes des utilisateurs et des groupes autorisés à accéder à une ressource particulière. La liste d'accès standard inclut les ID utilisateur et les noms de groupe autorisés à accéder à la ressource et le niveau d'accès octroyé à chacun d'eux.
- Définissez les profils des fonctions SCLM par rapport à la classe XFACILIT.
- L'administrateur de la sécurité définit les profils RACF requis à l'aide de la commande RDEFINE et en permet l'accès à l'aide de la commande PERMIT.

Profils de sécurité

L'administrateur de la sécurité définit les profils requis dans la classe XFACILIT à l'aide de la commande RDEFINE et en permet l'accès à l'aide de la commande PERMIT. Pour savoir quels profils doivent être définis, voir tableau 11, à la page 48.

Tableau 11. Profils de sécurité SCLMDT Developer Toolkit

Fonction	Profil XFACILIT	Droit d'accès requis
Génération	SCLM.BUILD.project.projdef.group.type.member	READ
Promotion	SCLM.PROMOTE.project.projdef.group.type.member	READ
Déploiement	SCLM.DEPLOY.server.application.node.cell.project.projdef.group.type	READ

La liste ci-après décrit la signification des différents noms de section de profil :

SCLM	Constante ; indique un profil de fonction SCLM
GENERER	Constante ; indique la fonction GENERER
REMONTER	Constante ; indique la fonction REMONTER
DEPLOYER	Constante ; indique la fonction DEPLOYER
projet	Nom du projet SCLM ou * pour tous les projets
définition de projet	Autre nom de projet (correspond au nom de projet par défaut) ou * pour tous les autres projets
Groupe	Groupe SCLM à générer/promouvoir/déployer ou * pour tous les groupes
Type	Type SCLM ou * pour tous les types
Membre	Membre SCLM à générer/promouvoir ou * pour tous les membres
Serveur	Serveur de déploiement cible (SERVER_NAME dans le script de déploiement Ant) ou * pour tous les serveurs
Application	Nom de l'application WAS cible (APPLICATION_NAME dans le script de déploiement Ant) ou * pour toutes les applications
Noeud	Nom du noeud WAS cible (NODE_NAME dans le script de déploiement Ant) ou * pour tous les noeuds
Cellule	Nom de la cellule WAS cible (CELL_NAME dans le script de déploiement Ant) ou * pour toutes les cellules

ID utilisateur de substitution

Les fonctions de génération, promotion et déploiement prennent en charge l'utilisation d'un ID utilisateur de substitution pour exécuter la fonction. S'il est activé, tous les appels autorisés vers la fonction entraîneront l'exécution de cette dernière avec les droits de l'ID utilisateur de substitution et non de l'ID utilisateur demandeur.

L'activation de l'ID utilisateur de substitution est spécifique au profil et contrôlée par la chaîne "SUID=userid" dans la zone APPLDATA du profil de sécurité, userid correspondant à l'ID utilisateur de substitution. Si cette chaîne est présente, l'ID utilisateur de substitution est utilisé ; si elle n'est pas présente, l'ID utilisateur du demandeur est utilisé.

Exemple : Génération

Cet exemple répertorie les définitions de produit de sécurité requises pour protéger la fonction de génération d'un projet donné. Le même exemple peut être utilisé pour sécuriser la fonction de promotion, en remplaçant la règle SCLM.BUILD.* par la règle SCLM.PROMOTE.*.

La définition de profil ci-après sécurise tous les membres du projet intégré TESTPROJ au niveau du groupe PROD. Un ID utilisateur de substitution est également défini.

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

Cet exemple définit un profil de génération SCLM où :

- Projet = TESTPROJ
- Autre définition de projet = TESTPROJ
- Groupe SCLM = PROD
- Type SCLM = tous les types
- Membre = tous les membres
- Accès universel = NONE (par défaut, personne n'est autorisé pour le profil)
- ID utilisateur de substitution = IBMUSER

Remarque : La commande SETROPTS met à jour les tables en mémoire de RACF et active donc la définition.

L'exemple ci-après illustre les droits de sécurité définis pour les utilisateurs (ou les groupes d'utilisateurs) du projet TESTPROJ de l'exemple précédent :

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

La commande PERMIT correspond au profil RDEFINE d'origine et autorise l'utilisateur USERID à générer tout membre du projet TESTPROJ et du groupe PROD. Comme un ID utilisateur de substitution est stocké dans la zone des données d'application (APPLDATA) du profil correspondant, la GENERATION est effectuée sous cet ID utilisateur de substitution (dans le cas présent).

Exemple : Déploiement

Vous trouverez ci-après un exemple de création de profil de déploiement.

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

Détails WAS :

- Nom du serveur = SERVERY
- Nom de l'application = TESTAPP
- Nom du noeud = NODE1
- Nom de la cellule = CELL1

Détails du projet SCLM :

- Projet = TESTPROJ
- Autre définition de projet = TESTPROJ
- Groupe SCLM = PROD
- Type SCLM = J2EEDEP

Autres informations :

- Accès universel = NONE (par défaut, personne n'est autorisé pour le profil)
- Aucun ID utilisateur de substitution

L'exemple ci-après illustre les droits de sécurité définis pour un groupe d'utilisateurs pour le profil de déploiement précédemment défini :

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

Cela correspond au profil RDEFINE d'origine et permet à tout utilisateur du groupe RACF J2EEGRP d'effectuer un déploiement sur le serveur ci-dessus et à partir des mêmes détails de projet SCLM.

Chapitre 5. Générations et promotions lancées via le service CRON

Bien que la plupart des générations et promotions soient lancées via le client Developer Toolkit, il existe une disposition permettant de configurer les fichiers de promotion et de génération dans le système de fichiers z/OS UNIX System Services et de lancer ces générations ou compilations via le service CRON (time) dans UNIX System Services.

A l'aide de cette méthode, le client SCLM Developer Toolkit Client n'est pas nécessaire car les paramètres de promotion et de génération associés sont lus à partir d'un fichier de configuration du système de fichiers z/OS UNIX System Services et transmis au composant Developer Toolkit Host pour le traitement SCLM.

Ci-dessous figure une description des exemples SCLM Developer Toolkit qui fournissent des générations et des promotions lancées par CRON. Ces exemples sont disponibles dans le répertoire des exemples de Developer for System z, /usr/lpp/rdz/samples/. Une copie, qui doit être personnalisée en fonction de vos besoins, a été placée dans /etc/rdz/sclmdt/script/ lors de la personnalisation de Developer for System z.

BWBCRON1

Cet exemple REXX appelle l'interface hôte SCLM Developer Toolkit et transmet les paramètres de fonction. La sortie du traitement de la fonction s'affiche par défaut sur STDOUT mais peut être redirigée vers un fichier ou un journal z/OS UNIX. L'exemple REXX devra être personnalisé comme indiqué dans ce dernier. Cet exemple REXX doit être exécuté en conjonction avec l'entrée de l'exemple BWBCRONB, dans le cas d'une génération, ou l'exemple BWBCRONP, dans le cas d'une promotion.

BWBCRONB

Cet exemple REXX configure la chaîne d'entrée de paramètre de génération transmise au module BWBCRON1. L'exemple requiert la personnalisation de l'utilisateur afin de mettre à jour tous les paramètres de génération requis.

BWBCRONP

Cet exemple REXX configure la chaîne d'entrée de paramètre de promotion transmise au module BWBCRON1. L'exemple requiert la personnalisation de l'utilisateur afin de mettre à jour tous les paramètres de promotion requis.

bwbsqld.rex

Exemple de script de génération JAVA SQLJ XML ANT

Spécifications de STEPLIB et PATH

Les variables PATH et STEPLIB dans le profil système (/etc/profile) ou le profil utilisateur (/u/userid/.profile) devront être configurées pour localiser les travaux CRON (\$PATH) ainsi que les modules de chargement SCLM Developer Toolkit (\$STEPLIB) si les modules de chargement SCLM Developer Toolkit ne résident pas dans le LINKLIST. Exemple :

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

Exécution des travaux de génération CRON

Une fois que les travaux CRON ont été ajoutés à la variable PATH, ils peuvent être exécutés en transmettant la sortie de parameter_exec à processing_exec. La sortie peut être envoyée dans un fichier journal de sortie.

Syntaxe

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

Le signe "|" correspond au symbole de barre verticale de z/OS UNIX.

Exemple d'appel

A l'aide des exemples de nom fourni, l'exécutable de génération CRON peut être appelé comme suit (\$ représente l'invite z/OS UNIX) :

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

Pour plus d'informations sur les services CRON disponibles et le format CRONTAB, voir *UNIX System Services Command Reference* (SA22-7802) et *UNIX System Services Planning* (GA22-7800).

Vous pouvez également utiliser la commande z/OS UNIX en ligne, **man** :

- man cron
- man crontab
- man at

Exemples de travaux de génération CRON

Cette section illustre les exemples de travaux BWBCRON1, BWBCRONB, BWBCRONP fournis dans la bibliothèque SFEKSAMV.

L'exemple suivant montre le code BWBCRON1.

```
/* REXX */
/* Customize STEPLIB, _SCLMDT_CONF_HOME and _SCLMDT_WORK_HOME */
/*
The STEPLIB should reflect the load libraries for
Rational Developer for System z.
If these datasets resides in the LINKLIST then set STEPLIB to '' .
*/
STEPLIB = 'FEK.SFEKLOAD:FEK.SFEKAUTH'
/*
The Environment variables _SCLMDT_CONF_HOME and _SCLMDT_WORK_HOME determines the HOME
directories where the configuration files and workarea reside for
SCLM Developer Toolkit. Refer to the Rational Developer for System z
configuration file /etc/rdz/rsed.envvars for the correct value.
*/
_SCLMDT_CONF_HOME = '/etc/rdz/sclmdt'
_SCLMDT_WORK_HOME = '/var/rdz'

/* SAMPLE USAGE */
COMMAND : BWBCRONB|BWBCRON1 > BWBCRONB.log
(passes build parameter list to BWBCRON1 & outputs to BWBCRONB.log)
/*
/* DO NOT ALTER BELOW */

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

CALL BWBINT

EXIT
```

Figure 18. BWBCRON1 - Exécutable de génération CRON

L'exemple suivant montre le code BWBCRONB.

```

/* REXX */
/* SAMPLE BUILD PARAMETER FILE USED FOR CRON INITIATED BUILDS */
/* Update Build parameters below */
/* if parameter required as Blank then set as '' */
FUNCTION = 'BUILD'
PROJECT = '' /* SCLM Project */
PROJDEF = '' /* Alt proj definition */
TYPE = '' /* SCLM Type */
MEMBER = '' /* SCLM Member name */
GROUP = '' /* SCLM Group */
GROUPBLD = '' /* Build at Group */
REPDGRP = '' /* Users Development group */
BLDREPT = 'Y' /* Generate Build report */
BLDLIST = 'Y' /* Generate List on error */
BLDMSG = 'Y' /* Generate Build Messages */
BLDScope = 'N' /* Build Scope E/L/N/S */
BLDMODE = 'C' /* Build Mode C/F/R/U */
BLDMSGDS = '' /* Message dataset */
BLDRPTDS = '' /* Report dataset */
BLDLSTDS = '' /* list dataset */
BLDEXTDS = '' /* Exit dataset */
SUBMIT = 'BATCH' /* Online or Batch */
/*
SI exécution en mode BATCH et JCL JOBCARD requis pour remplacer la valeur
par défaut, ajoutez jusqu'à 4 lignes JOBCARD.
Spécifiez au format LINE et incluez la variable LINECNT pour le
nombre de lignes.
*/
LINECNT = 2
LINE.1 = '//SCLMBLD JOB (XXX),SCLMBUILD,MSGCLASS=X,NOTIFY=&SYSUID,'
LINE.2 = '// CLASS=A,REGION=0M'

/* DO NOT ALTER PARM BUILD VARIABLE BELOW */
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
/* outputs parameter string as input to BWBCRON1 */
SAY PARM1
If (SUBMIT = 'BATCH') & (LINECNT > 0) then
  Do
    SAY '<JOBCARD>'
    Do i = 1 to LINECNT
      SAY LINE.i
    End
    SAY '</JOBCARD>'
  Fin

```

Figure 19. BWBCRONB - Fichier de paramètres de génération

L'exemple suivant montre le code BWBCRONP.

```

/* SAMPLE PROMOTE PARAMETER FILE USED FOR CRON INITIATED PROMOTES */
/* Update Promote parms below in quotes. */
/* if parameter required as Blank then set as '' */
FUNCTION = 'PROMOTE'
PROJECT = '' /* SCLM Project */
PROJDEF = '' /* Alt proj definition(opt) */
TYPE = '' /* SCLM Type */
MEMBER = '' /* SCLM Member name */
GROUP = '' /* SCLM Group */
GROUPPRM = '' /* Promote at Group (opt) */
REPDGRP = '' /* Users Development group */
PRMREPT = 'Y' /* Generate Promote report */
PRMSG = 'Y' /* Generate Promote Messages*/
PRMSCOPE = 'N' /* Promote Scope E/L/N/S */
PRMODE = 'C' /* Promote Mode C/F/R/U */
PRMSGDS = '' /* Message dataset */
PRMRPTDS = '' /* Report dataset */
PRMEXTDS = '' /* Exit dataset */
SUBMIT = 'BATCH' /* Online or Batch */
/*
  SI exécution en mode BATCH et JCL JOBCARD requis pour remplacer la valeur
  par défaut, ajoutez jusqu'à 4 lignes JOBCARD.
  Spécifiez au format LINE et incluez la variable LINECNT pour le
  nombre de lignes.
*/
LINECNT = 2
LINE.1 = '//SCLMBLD JOB (XXX),SCLMBUILD,MSGCLASS=X,NOTIFY=&SYSUID,'
LINE.2 = '// CLASS=A,REGION=0M'

/* DO NOT ALTER PARM PROMOTE VARIABLE BELOW */
PARM1 = 'SCLMFUNC='FUNCTION'&PROJECT='PROJECT'&PROJDEF='PROJDEF||',
        '&TYPE='TYPE'&MEMBER='MEMBER'&GROUP='GROUP'&GROUPPRM='GROUPPRM||',
        '&REPDGRP='REPDGRP'&PRMREPT='PRMREPT||',
        '&PRMSG='PRMSG'&PRMSCOPE='PRMSCOPE'&PRMODE='PRMODE||',
        '&PRMSGDS='PRMSGDS'&PRMRPTDS='PRMRPTDS'&PRMEXTDS='PRMEXTDS||',
        '&SUBMIT='SUBMIT
/* outputs parameter string as input to BWBCRON1 */
SAY PARM1
If (SUBMIT = 'BATCH') & (LINECNT > 0) then
  Do
    SAY '<JOBCARD>'
    Do i = 1 to LINECNT
      SAY LINE.i
    End
    SAY '</JOBCARD>'
  Fin

```

Figure 20. BWBCRONP - Fichier de paramètres de promotion

Annexe A. Présentation de SCLM

SCLM Developer Toolkit, une fonction d'IBM Rational Developer for System z, offre les moyens permettant de gérer et générer les applications distribuées écrites dans Eclipse, à l'aide de SCLM (Software Configuration and Library Manager), le système de gestion du code source IBM z/OS.

Les langages et outils utilisés par les utilisateurs de grands systèmes et de systèmes distribués diffèrent autant que les environnements qu'ils emploient. En identifiant et assimilant les principaux concepts de ces environnements, il est possible de les intégrer correctement dans une structure cohésive.

En termes de structure d'application, SCLM Developer Toolkit est constitué d'un ensemble de plug-ins Eclipse et du code hôte z/OS correspondant qui permet d'utiliser les transports HTTP et RSE. Au niveau opérations, un développeur Eclipse enregistre un projet d'espace de travail dans SCLM. Les fichiers du projet peuvent être ajoutés à un projet SCLM, extraits et restitués et éventuellement créés et déployés. Tous ces services sont pilotés via le menu Team Actions. Du point de vue des administrateurs SCLM, ces derniers peuvent créer des projets, des types, des langages et les traducteurs de compilations associés. Les fonctions, telles que les codes de modification et d'autorisation dépendent des besoins.

Concepts de SCLM

Du point de vue d'un développeur Java/J2EE, les concepts suivants décrivent SCLM :

Dénomination des fichiers

Le système de fichiers z/OS ne prend en charge que les noms de fichier de huit caractères. L'interface de SCLM Developer Toolkit fournit un service de traduction qui prend en charge les conventions des noms longs Java. Certains fichiers spécifiques à SCLM doivent respecter la restriction sur la règle de dénomination. Il s'agit principalement des restrictions sur la structure de l'ARCHDEF, décrite dans «Format d'ARCHDEF J2EE», à la page 10.

Type

Chaque fichier (ou membre selon la terminologie SCLM) stocké dans un projet SCLM est stocké dans un fichier. Ce dernier est identifié par le type SCLM. Le type fait partie du nom de fichier, constitué de SCLM Project.Group.Type pour SCLM, et associé à un langage. De nombreux types peuvent être définis dans un projet SCLM. Ces types permettent de stocker deux fichiers de même nom dans le même projet SCLM. Chaque projet peut contenir de nombreux types. En appliquant l'utilisation du type, il est possible de stocker plusieurs projets Eclipse dans le projet SCLM, bien que chaque projet IDE possède éventuellement un fichier .project et un fichier .classpath. Si nous n'isolons pas ces fichiers à l'aide du type, il n'existe qu'une seule copie de ces fichiers dans SCLM.

L'administrateur SCLM est chargé de définir les types SCLM. Lorsque vous partagez un projet à l'aide de SCLM, vous devez connaître le type à utiliser lors du stockage des objets dans SCLM.

Langage

Lorsque vous ajoutez un fichier à SCLM, vous devez le stocker avec une certaine définition de langage. L'administrateur SCLM est là aussi chargé de définir les langages. Cette définition contrôle le comportement de SCLM lorsque les fichiers sont transférés vers et depuis le système hôte. À l'aide de la définition de langage, il est possible de déterminer si un certain type de fichier est traduit en nom long, stocké sous forme d'objet binaire ou traduit en ASCII ou EBCDIC (codage z/OS natif). Par exemple, la définition de langage JAVABIN peut faire référence à un objet binaire traduit en nom long. Il est également possible de définir WEBHTML comme représentant un fichier texte traduit en nom long à stocker en ASCII. Le nombre de définitions de langage est défini par projet. Il est nécessaire de savoir quel langage utiliser pour s'assurer que le fichier est stocké correctement dans SCLM et qu'il peut en être extrait. Le langage définit également la manière dont SCLM crée un objet.

Propriétés de SCLM

Tout fichier sous le contrôle de SCLM est associé à un certain nombre de propriétés. Ces propriétés correspondent en fait au mécanisme de mappage entre le fichier IDE et les propriétés SCLM correspondantes. Lorsque des appels de service sont envoyés à SCLM, ces données sont lues pour formuler les paramètres de service appropriés. Vous pouvez afficher ces paramètres à partir du menu Propriétés lorsque vous sélectionnez un membre contrôlé par SCLM à partir d'Eclipse.

Structure d'un projet SCLM

Lorsque vous partagez un projet à l'aide de SCLM, vous devez également spécifier à quel groupe de développement vous appartenez. Les structures de projet SCLM se présentent sous forme d'arborescence.

À l'origine, le code est stocké au niveau DEVELOPPEMENT. Une fois qu'il a été créé et testé, il peut passer au niveau TEST. Enfin, une fois que les tests ont abouti, il peut passer au niveau PRODUCTION. Cela représente généralement le produit développé. Si un incident est détecté dans le code au niveau production, les fichiers à éditer pour résoudre l'incident sont copiés au niveau développement et la procédure de création est recommencée. Seule une partie du code qui constitue l'application est copiée au niveau développement. SCLM effectue le suivi des éléments qui compose la compilation et du niveau auquel ces éléments sont stockés.

Le niveau Développement peut comporter plusieurs groupes. Cela permet de séparer le code stocké au niveau développement. SCLM offre également des paramètres permettant de déterminer le comportement du code stocké dans les différents groupes de développement en terme de possibilités de promotion.

ARCHDEF

La structure du projet IDE se compose généralement d'un ou plusieurs projets IDE. En stockant chacun de ces projets IDE dans un type SCLM différent, cette structure est conservée. Le fichier ARCHDEF définit ensuite les fichiers qui constituent un projet IDE. Chaque projet SCLM peut posséder plusieurs ARCHDEF. Une ARCHDEF peut faire référence à d'autres ARCHDEF pour que cette structure à plusieurs projets IDE puisse être définie dans la procédure de création, l'ARCHDEF représentant le moyen principal de définir une liste de compilations pour SCLM. Une procédure de création en représente l'analogie la plus proche. L'ARCHDEF répertorie les fichiers qui constituent la compilation et spécifie un

script de création qui permet d'indiquer l'emplacement des classes ou des fichiers JAR externes. Pour plus d'informations, reportez-vous à la section relative au Guide d'utilisation, dans le système d'aide en ligne.

Concepts de JAVA/J2EE

Lorsqu'un projet IDE est créé dans l'espace de travail, un fichier de description de projet est automatiquement généré et stocké sous le nom **.project**. Ce document XML contient les descriptions des 'générateurs' ou 'natures' associés au projet. Les 'générateurs' sont des générateurs de projets incrémentiels qui créent un état de création en fonction du contenu du projet. Lorsque le contenu du projet est modifié, ce fichier est mis à jour. Les 'natures' définissent et gèrent l'association entre un projet spécifique et une fonction ou un plug-in particulier.

Le fichier **.classpath** décrit le chemin d'accès permettant de rechercher les classes et les fichiers jar externes auxquels le code source de votre projet IDE fait référence. La fonction équivalente lors d'une création via SCLM Developer est définie avec l'instruction **CLASSPATH_JARS** dans les scripts de création Ant. Cette instruction décrit le chemin d'accès sur l'hôte z/OS permettant de rechercher les classes et les fichiers jar externes auxquels le code source de votre projet IDE fait référence.

Les fichiers **.classpath** et **.project** sont tous deux utilisés pour préserver la configuration de votre projet IDE pour que ce dernier puisse être recréé dans un autre espace de travail. Pour cette raison, il est recommandé qu'ils soient tous deux restitués dans SCLM comme partie intégrante du projet IDE.

Le développement de projets présente un aspect important, surtout dans les projets J2EE : il est possible de créer différentes formes d'exécutable d'application. Les fichiers exécutables des projets Java sont souvent fournis sous forme de fichiers JAR, WAR, RAR ou EAR.

Les fichiers JAR sont généralement associés à des applications Java standard ou des EJB (Enterprise Java Bean).

Les fichiers WAR sont créés pour les applications Web. Ils sont en général composés de servlets Java, de fichiers JSP et de fichiers HTML. Les applications WAR représentent souvent applications le premier niveau des applications Web. Ces applications peuvent faire référence à d'autres fichiers JAR, tels que des EJB pour des services spécifiques. Chaque fichier WAR contient un fichier **web.xml**. Ce fichier décrit la composition de l'application WAR en termes de Java, HTML et de services de bibliothèque qu'il utilise.

Le développement des fichiers RAR n'est actuellement pas pris en charge dans SCLM Developer Toolkit.

Les fichiers EAR représentent les applications d'entreprise. Ces applications sont composées de fichiers JAR et WAR. En langage J2EE, la création du fichier EAR correspond à l'assemblage de ses fichiers JAR et WAR constituants. Cette méthode d'assemblage permet de créer des applications EAR qui sont en fait constituées de composants spécifiques (JAR/WAR). La véritable composition de l'application est décrite dans le fichier **application.xml**. Le fichier EAR lui-même n'est pas un objet exécutable autonome. Il doit être installé dans un conteneur J2EE tel que WAS (Websphere Application Server). L'installation du fichier EAR est appelé déploiement. Il est possible d'accéder à une application EAR déployée via l'environnement WAS. Le déploiement est une procédure distincte de la création. Il implique l'installation physique de l'application EAR.

Le développement d'applications J2EE risque donc d'impliquer le développement d'un certain nombre de composants distincts, tels que des fichiers WAR et JAR. Ces fichiers sont ensuite assemblés dans un fichier EAR. Le fichier EAR est alors prêt à être déployé dans un conteneur J2EE (par exemple, WAS) pour être utilisé.

Dans l'espace de travail Eclipse, les projets sont en réalité assez proches ; dans l'environnement IDE, ils peuvent en fait faire facilement référence à d'autres projets IDE en termes de conditionnement. Dans SCLM, chacun de ces projets IDE (par exemple, les projets WAR, JAR et EAR) doivent être mappés dans un même projet SCLM, chaque projet étant différencié via l'utilisation d'un type SCLM différent. En effet, des noms de fichier communs étant utilisés dans de nombreux projets IDE, tels que des fichiers .project, .classpath, web.xml et application.xml, l'utilisation de types distincts permet à des composants de même nom de coexister dans un même projet SCLM. Pour plus d'informations sur le mappage, voir «Mappage de projets J2EE à SCLM», à la page 18.

Du point de vue de SCLM, il est préférable de référencer le développement de l'application EAR par l'intermédiaire d'une structure ARCHDEF de haut niveau. Dans SCLM, les ARCHDEF de haut niveau, appelés *package* dans de nombreux projets SCLM, représentent l'apex de la structure des ARCHDEF, qui est suivi de l'application EAR et des références de niveau inférieur (fichiers WAR et JAR) qui constituent l'application EAR. Cette structure permet l'utilisation de compilations aux niveaux supérieurs et inférieurs, ainsi que l'utilisation de compilations intégrales ou conditionnelles. Les ARCHDEF permettent donc également de définir les éléments du projet J2EE dans le projet SCLM.

Annexe B. Table de traduction des noms longs/abrégés

Actuellement, le noyau de SCLM ne prend pas en charge l'utilisation du stockage du code avec des noms supérieurs à huit caractères.

Les codes, tels que les codes Java et les autres codes de client PC, possèdent de par leur nature des noms de longueur bien supérieure et incorporent même des informations sur le chemin d'accès (conditionnement) dans leur nom. Ainsi, le code dont certaines parties possèdent des noms de plus de huit caractères doit passer par un utilitaire de conversion de nom long/abrégé pour que ces parties puissent être stockées dans SCLM sous un nom de huit caractères ou moins.

Une table de traduction des noms longs en noms abrégés stocke les noms longs (noms réels) correspondant aux noms abrégés (tels qu'ils sont stockés dans SCLM). SCLM accède à ces tables, qui sont sauvegardées dans un fichier VSAM, et les contrôle. Cette fonction est apparue dans SCLM avec la PTF de l'APAR OA11426 pour z/OS 1.4 et les versions ultérieures. Pour z/OS 1.8 et les versions ultérieures, cette PTF n'est pas requise.

L'algorithme de conversion suit les étapes suivantes :

1. Le préfixe de traduction est composé des deux premiers caractères (en majuscules) du nom de fichier/programme long (à savoir, le dernier nom de fichier après le caractère "/" dans un format à plusieurs conditionnements). Si ces deux premiers caractères ne sont pas valides comme préfixe de nom de membre hôte (car ils contiennent des caractères spéciaux non valides), le préfixe "XX" est utilisé. Pour les cas spéciaux, tels qu'un nom alphabétique à un caractère (/u/test/A ou /u/test/A.java) le préfixe "XX" est également affecté.
2. Les six derniers caractères sont numériques et correspondent au prochain nombre séquentiel disponible dans la table de traduction.

Exemple

Nom long	Nom abrégé dans le fichier partitionné ou le fichier partitionné étendu SCLM
com/ibm/workbench/testprogram.java	TE000001
source/plugins/Phantom/.classpath	XX000001

Récapitulatif technique du programme de traduction SCLM

Le programme SCLM FLMLSTRN a été créé pour lire et mettre à jour la table de traduction VSAM. SCLM Developer Toolkit utilise ce programme pour lire et mettre à jour les noms longs et les noms abrégés en corrélation.

Le fichier VSAM permettant de stocker la table de traduction est un fichier KSDS de longueur variable pour lequel un chemin d'accès et un index secondaire sont définis. Un exemple de travail est fourni dans SCLM pour allouer ce fichier VSAM.

La structure interne du cluster VSAM est la suivante :

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

Vous trouverez un exemple de JCL permettant d'allouer le fichier VSAM de traduction des noms longs/abrégés à l'étape 6, relative à la configuration du fichier VSAM de traduction des noms longs/abrégés.

Remarque : Les informations techniques ci-après sur les appels de fonction de la table de traduction SCLM ne sont fournies qu'à titre d'informations et ne sont pas requises pour activer une fonctionnalité SCLM Developer Toolkit.

Le programme FLMLSTRN est appelé à l'aide du service ISPF SELECT avec l'un des paramètres répertoriés dans le tableau 12.

Syntaxe :

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

Exemple d'appel :

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

Tableau 12. Paramètres de traduction des noms longs/abrégés.

Enregistrement du mot clé	Traitement	Description
FINDLONG	Unique	Recherche un nom long pour un nom abrégé donné
FINDSHORT	Unique	Recherche un nom abrégé pour un nom long donné
TRANSLATE	Unique	Recherche un nom abrégé s'il existe ou affecte un nouveau nom abrégé dans le cas contraire
MIGRATE	Multiple	Recherche plusieurs noms longs
IMPORT	Multiple	Recherche plusieurs noms abrégés

Traitement unique des enregistrements de nom long/abrégé

Traitement FINDLONG

- Le cluster VSAM alloué à DD LSTRANS est ouvert en mode lecture.
- Le nom abrégé est extrait de la variable ISPF FLMLSSHR et ce nom abrégé est utilisé pour lire le fichier VSAM.
- Si l'enregistrement est introuvable, un message est renvoyé via la variable ISPF FLMLSERR pour indiquer que le nom long est introuvable.
- Si le nom long est détecté, il est renvoyé dans la variable ISPF FLMLSLNG.
- Le cluster VSAM est fermé.

Traitement FINDSHORT

- Le chemin d'accès VSAM alloué à DD LSTRNPTH est ouvert en mode lecture.
- Le nom long est extrait de la variable ISPF FLMLSLNG.

- Les derniers 50 octets du nom long permettent de lire le chemin d'accès.
- Si aucun enregistrement n'est renvoyé, un message est renvoyé via la variable ISPF FLMLSERR pour indiquer que le nom abrégé est introuvable.
- Si un enregistrement est renvoyé, le nom long de l'enregistrement VSAM est comparé au nom long de la variable ISPF FLMLSLNG.
- S'il ne correspond pas, les enregistrements VSAM sont lus et comparés jusqu'à ce que la clé ls_lngname_key ne corresponde pas ou que le nom long soit détecté.

Remarque : La clé ls_lngname_key autorise les doublons car un enregistrement VSAM peut posséder une même clé ls_lngname_key, mais un nom long différent.

- Si le nom abrégé est détecté, il est renvoyé dans la variable ISPF FLMLSSHR.
- Le chemin d'accès de VSAM est fermé.

Traitement TRANSLATE

Le traitement est identique à celui de FINDSHORT :

- Si le nom abrégé est détecté, aucun autre traitement n'est effectué.
- Si le nom abrégé est introuvable, le cluster VSAM alloué à DD LSTRANS est ouvert en mode mise à jour.
- Le nom de fichier est déterminé en recherchant le dernier caractère '/' ou '\' dans le nom long.
- Les deux premiers octets du nom de fichier permettent de rechercher l'enregistrement du préfixe du fichier VSAM, qui contient un numéro.
- Le préfixe du fichier et son numéro permettent de générer le nom abrégé (par exemple, PR000123).
- Le nom abrégé généré (PR000123) permet de vérifier le fichier VSAM pour déterminer si le nom abrégé est utilisé.
- Si c'est le cas, le numéro du préfixe est incrémenté et le nom abrégé est de nouveau vérifié.
- Cette procédure se poursuit jusqu'à ce qu'un nom abrégé non utilisé soit détecté.
- L'enregistrement du préfixe est mis à jour, puis le nouvel enregistrement de traduction est ajouté.
- Le nom abrégé est renvoyé dans la variable ISPF FLMLSSHR.
- Le cluster VSAM est fermé.

Traitement multiple des enregistrements de nom long/abrégé

Les fonctions MIGRATE et IMPORT ont été introduites pour améliorer les performances lors de la traduction d'un nombre important de noms longs traduits (MIGRATE) ou de la recherche d'un nombre important de noms abrégés (IMPORT).

Ces deux fonctions, "MIGRATE" et "IMPORT", lisent un fichier séquentiel bloqué par variable avec LRECL=1036, qui est alloué comme DD LSTRNPRC.

Avant d'être appelé, ce fichier contient les noms abrégés ou les noms longs, suivant la fonction appelée, dans la colonne et au format appropriés.

Après avoir été appelé, LSTRNPRC contient le nom abrégé et le nom long associé.

Le format du fichier est le suivant :

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

Traitement IMPORT

- Le cluster VSAM alloué à DD LSTRANS est ouvert en mode lecture et le fichier de traitement alloué à DD LSTRNPRC est ouvert pour mise à jour.
- Pour chacun des enregistrements du fichier de traitement, le nom abrégé est utilisé pour lire le fichier de traduction VSAM. Si un enregistrement est détecté, le fichier de traitement est mis à jour avec le nom long.
- Les fichiers de traitement/clusters VSAM sont fermés.

Traitement MIGRATE

- Le cluster VSAM alloué à DD LSTRANS est ouvert en mode lecture et le fichier de traitement alloué à DD LSTRNPRC est ouvert pour mise à jour.
- Pour chacun des enregistrements du fichier de traitement, le nom long est utilisé pour lire le fichier VSAM. Si un enregistrement est détecté, l'enregistrement du fichier de traitement est mis à jour avec son nom abrégé correspondant ; sinon, LSTRANS est ouvert en mode mise à jour pour ajouter les nouvelles entrées de nom long/abrégé et le nouveau nom abrégé généré est réenregistré dans le fichier LSTRNPRC.
- Les fichiers de traitement/clusters VSAM sont fermés.

Voici un exemple de code REXX permettant d'appeler la procédure de traduction des noms longs/abrégés.

```
/* REXX *****/  
/* Sample to translate long name to a short name */  
/* *****/  
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"  
/* Create short name for long name 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  
End  
Else  
Do  
    Address ISPEXEC "VGET (FLMLSSHR,FLMLSLNG) PROFILE"  
    Say " Shortname = " FLMLSSHR  
    Say " Longname = " FLMLSLNG  
End  
Address TSO  
"FREE FI(LSTRANS)"  
"FREE FI(LSTRNPTH)"
```

Figure 21. Exemple de code REXX permettant d'appeler le module de traduction

Annexe C. API de SCLM Developer Toolkit

Cette annexe présente l'API des services hôte de SCLM Developer Toolkit. Elle utilise un format de demandes et réponses basé XML et est accessible via le système de fichiers z/OS UNIX Systems Services.

Cette API permet aux utilisateurs d'utiliser les services hôte de SCLM Developer Toolkit avec leurs propres client/plug-in et couche de transport s'ils le souhaitent.

Un exemple de programme Java (avec entrée et sortie) est fourni, qui permet d'accéder à l'API des services hôte de SCLMDT à l'aide d'un serveur HTTP comme mécanisme de transport.

La plupart des demandes de fonction reposent sur les services hôte SCLM actuels et vous trouverez des informations supplémentaires sur les valeurs de paramètre similaires pour les fonctions communes dans le guide SCLM et le guide de référence de la version z/OS appropriée.

Remarque : Cette API aborde les services hôte actuellement utilisés par SCLM Developer Toolkit Client. Par conséquent, de nombreuses fonctions risquent de ne pas pouvoir être aussi bien utilisées par les clients ou nécessitent une validation supplémentaire côté client.

Format d'appel

L'exemple de commande suivante illustre comment les services hôte de SCLMDT peuvent être appelés :

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

cat	Commande z/OS UNIX permettant d'afficher les fichiers texte
sclmdt_request.xml	Fichier XML en entrée avec la demande utilisateur
	Commande z/OS UNIX permettant de transférer la sortie de la commande précédente comme entrée de la commande suivante
BWBXML	Script de conversion XML, qui se trouve dans le répertoire /bin de Developer for System z, qui appelle l'interface de service SCLMDT
>	Commande z/OS UNIX permettant de rediriger la sortie de la commande précédente vers un fichier
sclmdt_response.xml	Fichier XML en sortie, contenant la réponse du service

Remarques :

1. La variable d'environnement PATH doit contenir l'emplacement du répertoire de BWBXML. Par exemple :

```
export PATH=/usr/lpp/rdz/bin:$PATH
```
2. Si HTTP est utilisé comme mécanisme de transport :
 - Le script d'interface BWBXML peut être appelé comme script CGI (type EXEC dans les instructions du serveur HTTP).
 - La demande est lue comme une entrée standard (STDIN) via BWBXML et peut donc être transmise au script CGI comme demande POST (voir l'exemple de programme).

Schéma XML des commandes SCLMDT

L'exemple ci-après montre le schéma XML des commandes SCLMDT référencées dans le fichier en entrée XML. Cet exemple est également disponible comme membre BWBXSD1 dans l'exemple de bibliothèque 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">
<!-- Specifies native TSO or ISPF service call -->
<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">
<!-- Default NONE : Session terminates after service call-->
<xs:enumeration value="NONE"/>
<!-- Reuseable ISPF session stays active between calls -->
<xs:enumeration value="REUSE"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<!-- Use existing ISPF profile in call -->
<xs:element name="ispprof" type="xs:string" minOccurs="0"/>
<!-- Free form TSO/ISPF command -->
<xs:element name="command" type="xs:string" minOccurs="0"/>
<!-- List of all SCLM DT functions available -->
<!-- sclmfunc : SCLM function selected -->
<xs:element name="sclmfunc" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="BUILD"/> <!-- SCLM build function-->
<!-- Build parms : project,projdef,group,repdgrp,type,member,bldmode,bldlist,
bldrept,bldmsg,bldmsgds,bldlist,bldlstds,bldextds,groupbld,submit -->
<xs:enumeration value="PROMOTE"/> <!-- SCLM promote function -->
<!-- Promote parms : project,projdef,group,repdgrp,type,member,prmmode,prmrept,
prmmgs,prmsgds,prmxtds,groupprm,submit -->
<xs:enumeration value="EDIT"/> <!-- EDIT member -->
<!-- Edit parms : project,projdef,group,repdgrp,type,member,lang -->
<!-- Note: member transferred to DTinstall/WORKAREA -->
<xs:enumeration value="BROWSE"/> <!-- Browse member -->
<!-- Browse parms : project,projdef,group,repdgrp,type,member,lang -->
<!-- Note: member transferred to DTinstall/WORKAREA -->
<xs:enumeration value="SAVE"/> <!-- Save edited member-->
<!-- Save parms : project,projdef,group,repdgrp,type,member,lang -->
<!-- Note: member received from DTinstall/WORKAREA -->
<xs:enumeration value="DELETE"/> <!-- SCLM delete member -->
<!-- Delete parms : project,projdef,group,repdgrp,type,member,delflag -->
<xs:enumeration value="UNLOCK"/> <!--
<!-- Unlock parms : project,projdef,group,repdgrp,type,member -->
<xs:enumeration value="DEPLOY"/> <!-- J2EE deploy function -->
<!-- Deploy parms : project,projdef,group,repdgrp,type,member,
depmode,depsec,groupdpy -->
<xs:enumeration value="REPORT"/> <!-- SCLM project list -->
<!-- Report parms : project,projdef,reptype,repdgrp,repccode,repLang,
repacode,repmem,repgrp,repmode,repbmap -->
<xs:enumeration value="MIGDSN"/> <!-- List non-sclm datasets & members -->
<!-- Migdsn parms : project,projdef,groupdev,migmemb,migdsn -->
<xs:enumeration value="MIGPDS"/> <!-- Migrate NON-SCLM datasets to SCLM -->
<!-- Migpds parms : project,projdef,group,type,migfile,migmode,
authcode,ccode,migonly -->
<xs:enumeration value="INFO"/> <!-- SCLM member status information -->
<!-- Info parms : project,projdef,group,repdgrp,type,member,lang -->
<xs:enumeration value="UPDATE"/> <!-- update SCLM member information -->
<!-- Update parms : project,projdef,group,repdgrp,type,member,
lang,ccode,authcode -->
<xs:enumeration value="AUTHUPD"/> <!-- Change SCLM member authcode -->
<!-- Authupd parms : project,projdef,group,type,member,authcode -->
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:all>
</xs:complexType>
</xs:element>
</xs:schema>
```

Paramètres et fonctions des demandes

Format des fonctions

L'exemple ci-après montre la structure de base du fichier en entrée XML, #function représentant la fonction appelée et #parameter et #value, un paramètre et sa valeur.

```
<?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>
```

Figure 23. Structure de base du fichier en entrée XML

Remarque : Si un paramètre est spécifié plusieurs fois, les définitions de la dernière instance seront utilisées.

La liste suivante représente des remarques générales sur les paramètres et la manière dont ils sont abordés dans la présente publication :

- Les paramètres ne sont pas positionnels.
- Les crochets ([]) désignent un paramètre facultatif de la fonction.
- Les accolades ({}) désignent une liste de valeurs possibles pour le paramètre.
- Une barre verticale (|) sépare les valeurs multiples dans une liste. La valeur soulignée, s'il en existe une, correspond à la valeur par défaut.
- Les mots clés en majuscules représentent des constantes qui doivent être codées telles quelles ; ceux en minuscules, des marques de réservation à remplacer par des valeurs personnalisées.
- Chaque paramètre requiert une valeur, mais seules les valeurs de la liste sont présentées.
- Les références à WORKAREA font référence au répertoire WORKAREA de z/OS UNIX, qui se trouve par défaut dans /var/rdz/sclmdt/.
- Les références à member requièrent l'utilisation du nom de membre abrégé (SCLM), sauf indication contraire.

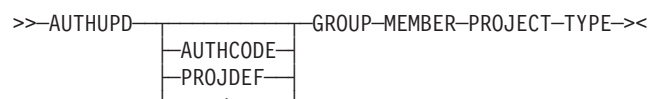
Liste des fonctions

- «AUTHUPD – Modification du code des droits d'accès SCLM», à la page 69
- «BROWSE – Exploration d'un membre SCLM», à la page 70
- «BUILD – Création d'un membre SCLM», à la page 70
- «DELETE – Suppression d'un membre SCLM», à la page 72
- «DEPLOY – Déploiement d'un fichier EAR J2EE», à la page 72
- «EDIT – Edition d'un membre SCLM», à la page 73
- «INFO – Informations sur le statut du membre SCLM», à la page 74
- «J2EEIMP – Importation d'un projet à partir de SCLM», à la page 74

- «J2EEMIG – Migration d'un projet dans SCLM», à la page 75
- «J2EEMIGB – Migration par lots d'un projet dans SCLM», à la page 77
- «JARCOPY – Copie d'un fichier JAR», à la page 77
- «JOBSTAT – Extraction du statut d'un travail par lots», à la page 78
- «LRECL – Extraction du fichier SCLM ou LRECL», à la page 78
- «MIGDSN – Liste des membres et des fichiers non SCLM», à la page 79
- «MIGPDS – Migration des membres et des fichiers NON-SCLM dans SCLM», à la page 79
- «PROJGRPS – Extraction des groupes SCLM d'un projet», à la page 80
- «PROJINFO – Extraction des informations sur un projet SCLM», à la page 80
- «PROMOTE – Promotion d'un membre SCLM», à la page 80
- «REPORT – Création d'un rapport de projet», à la page 82
- «REPUTIL –Rapport SCLM DBUTIL», à la page 82
- «SAVE – Sauvegarde d'un membre SCLM», à la page 83
- «UNLOCK – Déverrouillage d'un membre SCLM», à la page 84
- «UPDATE – Mise à jour des informations d'un membre SCLM», à la page 84
- «VERBROW – Versions d'exploration de SCLM», à la page 85
- «VERDEL – Suppression de versions SCLM», à la page 85
- «VERHIST – Historique des versions SCLM», à la page 86
- «VERLIST – Versions des listes SCLM», à la page 86
- «VERREC – Versions de restauration SCLM», à la page 87

AUTHUPD – Modification du code des droits d'accès SCLM

Cette fonction modifie le code des droits d'accès d'un membre.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[AUTHCODE]

Code des droits d'accès SCLM - Nouveau code des droits d'accès à affecter à un membre.

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

BROWSE – Exploration d'un membre SCLM

Cette fonction copie un membre de SCLM dans le répertoire z/OS UNIX WORKAREA/userid/EDIT/ . Aucun verrouillage d'édition n'est effectué dans SCLM.

>>BROWSE

PROJDEF

 GROUP-MEMBER-PROJECT-TYPE-><

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

BUILD – Création d'un membre SCLM

Cette fonction demande à SCLM de compiler, d'associer et d'intégrer les composants logiciels aux définitions de l'architecture du projet.

>>BUILD

BLDEXTDS
BLDLIST
BLDLSTD
BLDMODE
BLDMSGDS
BLDREPT
BLDRPTDS
BLDScope
PROJDEF
SUBMIT

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

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

GROUPBLD

Groupe SCLM - Groupe SCLM dans lequel le membre requis doit être créé.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[BLDEXTDS [dsn | NONE]]

Fichier d'exit de création - NONE ou nom du fichier devant contenir la sortie de l'exit de création, si un exit de création est utilisé. Si le fichier n'existe pas, un nouveau fichier est alloué. La valeur par défaut est NONE.

[BLDLIST [Y | N]]

Liste de compilations - Indique si les listes de traduction des compilations ne doivent être copiées dans le fichier de liste qu'en cas d'erreur. La valeur par défaut est Y.

[BLDLSTDS [dsn | NONE]]

Fichier des listes de compilations - NONE ou nom du fichier devant contenir les listes de compilations. Si le fichier n'existe pas, un nouveau fichier est alloué. Les listes de compilation sont également renvoyées dans le fichier de réponses XML, quel que soit la valeur de ce paramètre. La valeur par défaut est NONE.

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

Mode de compilation - Indique le mode de compilation (C=conditionnel, F=Forcé, R=rapport, U=inconditionnel). La valeur par défaut est C.

[BLDMSGDS [dsn | NONE]]

Fichier des messages de compilation - NONE ou nom du fichier devant contenir les messages de compilation. Si le fichier n'existe pas, un nouveau fichier est alloué. Les messages de compilation sont également renvoyés dans le fichier de réponses XML, quel que soit la valeur de ce paramètre. La valeur par défaut est NONE.

[BLDREPT [Y | N]]

Rapport de création - Indique si un rapport de création doit être généré. La valeur par défaut est Y.

[BLDRPTDS [dsn | NONE]]

Fichier du rapport de création - NONE ou nom du fichier devant contenir le rapport de création. Si le fichier n'existe pas, un nouveau fichier est alloué. Le rapport de compilation est également renvoyé dans le fichier de réponses XML, quel que soit la valeur de ce paramètre. La valeur par défaut est NONE.

[BLDSCOPE [E | L | N | S]]

Portée de la création - Indique la portée de la création (E=étendue, L=limitée, N=normale, S=sous-unité). La valeur par défaut est N.

[PROJDEF]

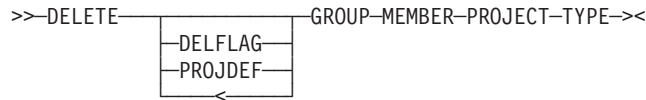
Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[SUBMIT [BATCH | ONLINE]]

Méthode de soumission - La création est soumise en ligne ou dans un traitement par lots. La valeur par défaut est ONLINE.

DELETE – Suppression d’un membre SCLM

Cette fonction supprime un membre SCLM.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

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

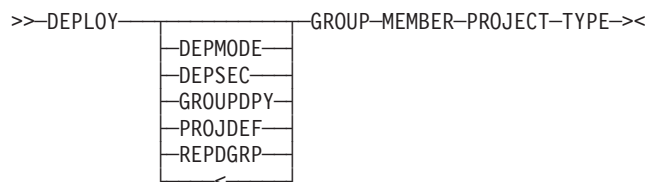
Option de suppression - Indique que le texte, le compte, la mappe de création, une combinaison de la mappe de création et du texte, ou le tout doit être supprimé pour un membre donné. La valeur par défaut est ALL.

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

DEPLOY – Déploiement d’un fichier EAR J2EE

La fonction DEPLOY exécute le script de déploiement référencé par le membre pour déployer un fichier d’archive d’entreprise (EAR) J2EE d’USS ou de SCLM dans WAS (Websphere Application Server). Pour plus d’informations sur la création d’un membre de script de déploiement, reportez-vous au Guide d’utilisation de SCLM Developer Toolkit.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[DEPSEC {Y | N}]

Sécurisation du déploiement - Option indiquant si une vérification de la règle de sécurité est effectuée pour ce déploiement et si l'utilisation d'un ID utilisateur de substitution est possible.

[DEPMODE {R}]

Mode de déploiement - Mode rapport uniquement, si la valeur 'R' est spécifiée. Aucun déploiement n'est effectué.

[GROUPDPY]

Groupe de déploiement - Groupe SCLM à partir duquel le fichier EAR sera déployé (ou hiérarchie de recherche des groupes, si aucun groupe n'est détecté). Si la valeur spécifiée est 'USS', le fichier EAR est déployé directement à partir du répertoire z/OS UNIX indiqué dans le nom EAR, dans la variable EAR_FILE_NAME. Cette variable est définie dans le membre de script de déploiement référencé sous 'MEMBER'.

[PROJDEF]

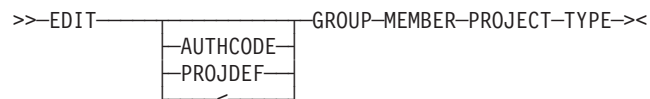
Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[REPDGRP]

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

EDIT – Edition d'un membre SCLM

Cette fonction copie un membre de SCLM dans le répertoire z/OS UNIX WORKAREA/userid/EDIT/. Elle crée également un verrou SCLM sur le membre avec l'ID utilisateur comme clé d'accès.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[AUTHCODE]

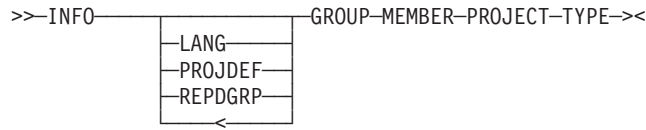
Code des droits d'accès SCLM - Nouveau code des droits d'accès à affecter à un membre.

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

INFO – Informations sur le statut du membre SCLM

Cette fonction fournit des informations sur le statut du membre SCLM.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[LANG]

Langage SCLM - Langage SCLM du membre sélectionné.

[PROJDEF]

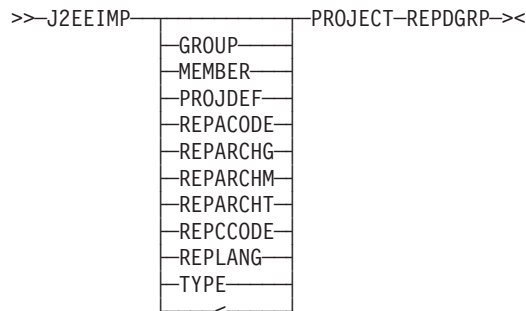
Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[REPDGRP]

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

J2EEIMP – Importation d'un projet à partir de SCLM

Cette fonction importe un projet de SCLM dans le répertoire z/OS UNIX /var/rdz/WORKAREA/userid au format JAR (zippé). Le fichier du projet JAR peut alors être copié sur le client. Le nom du fichier du projet est renvoyé dans le mot clé J2EEFILE de la sortie de la fonction (XML).



Paramètres requis :

[PROJECT]

Nom du projet SCLM - Nom du projet SCLM.

[REPDGRP]

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

Paramètres facultatifs :

[GROUP {group | group* | *}]

Hiérarchie des groupes - Hiérarchie des groupes pour la sélection des membres. Si REPARCHM est défini, les membres associés à cette définition d'archive se trouvent dans group*.

[MEMBER {member | member* | *}]

Membre SCLM - Membre(s) SCLM sélectionné(s).

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[REPACODE]

Code des droits d'accès - Code des droits d'accès cible du filtrage.

[REPARCHG]

Groupe d'Archdef - Groupe SCLM dans lequel se trouve le membre d'Archdef.

[REPARCHM]

Membre d'Archdef - Nom du membre d'Archdef à sélectionner pour l'importation.

[REPARCHT]

Type d'Archdef - Type SCLM dans lequel se trouve le membre d'Archdef.

[REPCCODE]

Code de modification - Code de modification cible du filtrage.

[REPLANG]

Langage SCLM - Langage cible du filtrage.

[TYPE {type | type* | *}]

Type SCLM - Type(s) SCLM permettant de sélectionner les membres.

J2EEMIG – Migration d'un projet dans SCLM

Cette fonction extrait un fichier compressé (au format JAR) du répertoire USS et migre tous les membres de ce fichier dans SCLM, en convertissant le nom long en nom abrégé si nécessaire.

```
>>-J2EEMIG-

|          |
|----------|
| ARCHAC   |
| ARCHCC   |
| ARCHTYPE |
| AUTHCODE |
| CCODE    |
| J2EESINC |
| J2EETYPE |
| MIGMODE  |
| PROJARCH |
| PROJDEF  |
| SCLMREFS |
| SUBMIT   |

-<GROUP-J2EEFILE-LANG-MEMBER-PROJECT-TYPE-><
```

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

J2EEFILE

Nom du fichier en entrée - Nom de fichier du projet JAR (zippé) à importer dans SCLM. Le fichier JAR doit se trouver dans le répertoire z/OS UNIX /var/rdz/WORKAREA/userid.

LANG

Langage SCLM - Langage SCLM du membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[ARCHAC]

Code des droits d'accès de l'Archdef - Définit le code des droits d'accès de l'Archdef.

[ARCHCC]

Code de modification de l'Archdef - Définit le code de modification de l'Archdef.

[ARCHTYPE [ARCHDEF | archtype]]

Type SCLM de l'Archdef - Définit le type de l'Archdef.

[AUTHCODE]

Code des droits d'accès des membres - Définit le code des droits d'accès des membres de l'Archdef.

[CCODE]

Code de modification des membres - Définit le code de modification des membres de l'Archdef.

[J2EESINC]

J2EE SINC Build Scrip. - Nom du script de génération qui se trouve dans TYPE J2EEBLD et qui est référencé par le mot clé SINC dans le membre ARCHDEF.

[J2EETYPE {JAR | WAR | EAR}]

Type J2EE - Spécifie JAR, WAR ou EAR pour le projet d'Archdef J2EE.

[MIGMODE {FORCE}]

Mode de migration – FORCE remplace les membres existants. La valeur par défaut est une migration conditionnelle.

[PROJARCH]

Projet de l'Archdef - Nom de l'Archdef à mettre à jour ou à créer

[SCLMREFS]

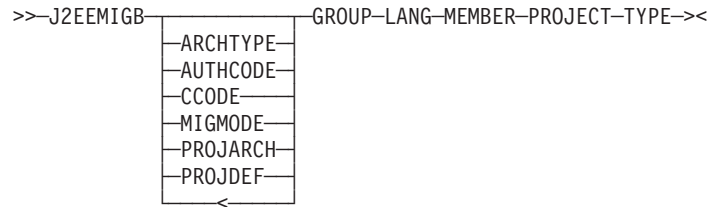
Références SCLM - Autres Archdef ou composants.

[SUBMIT [BATCH | ONLINE]]

Méthode de soumission - La migration est soumise en ligne ou dans un traitement par lots. La valeur par défaut est ONLINE.

J2EEMIGB – Migration par lots d'un projet dans SCLM

Cette fonction configure et exécute le travail PAR LOTS de la migration.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

LANG

Langage SCLM - Langage SCLM du membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[ARCHTYPE]

Type SCLM de l'Archdef - Définit le type de l'Archdef.

[AUTHCODE]

Code des droits d'accès des membres - Définit le code des droits d'accès des membres de l'Archdef.

[CCODE]

Code de modification des membres - Définit le code de modification des membres de l'Archdef.

[MIGMODE {FORCE}]

Mode de migration – FORCE remplace les membres existants. La valeur par défaut est une migration conditionnelle.

[PROJARCH]

Projet de l'Archdef - Nom de l'Archdef à mettre à jour ou à créer.

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

JARCOPY – Copie d'un fichier JAR

Cette fonction copie un fichier JAR stocké dans SCLM dans un répertoire z/OS UNIX, qui peut être utilisé comme répertoire CLASSPATH.



Paramètres requis :

CLASSDIR

Répertoire du chemin d'accès aux classes - Nom du répertoire z/OS UNIX cible.

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

JARNAME

Nom du fichier JAR - Nom long du fichier JAR cible. Le nom abrégé SCLM est recherché automatiquement.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

JOBSTAT – Extraction du statut d'un travail par lots

Cette fonction extrait le statut d'un travail par lots spécifique.

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

Paramètres :

JOBFUNC {JOBSTAT | JOBRETR}

Sélection d'une fonction - Extrait le statut du travail (JOBSTAT) ou la sortie du travail (JOBRETR).

JOBID

ID travail - Numéro de travail du travail par lots.

JOBNAME

Nom du travail - Nom du travail par lots.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

LRECL – Extraction du fichier SCLM ou LRECL

Cette fonction extrait la longueur d'enregistrement logique d'un fichier SCLM.

>>LRECL—GROUP—PROJECT—REPDGRP—TYPE—<<

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

MIGDSN – Liste des membres et des fichiers non SCLM

Cette fonction répertorie les membres et les fichiers sélectionnés qui ne sont pas gérés par SCLM (pour une migration ultérieure dans SCLM).

```
>>MIGDSN_____MIGDSN-MIGMEM-PROJECT-><
      |__PROJDEF__|
```

Paramètres requis :

MIGDSN [dsn | *]

Filtrage des fichiers - Filtrage des fichiers pour la liste. La valeur par défaut est *.

MIGMEM [member | *]

Filtrage des membres - Filtrage des membres pour la liste. La valeur par défaut est *.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

MIGPDS – Migration des membres et des fichiers NON-SCLM dans SCLM

Cette fonction migre les membres et les fichiers sélectionnés qui ne sont pas gérés par SCLM dans SCLM.

```
>>MIGPDS_____GROUP-PROJECT-TYPE-><
      |__PROJDEF__|
```

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

PROJGRPS – Extraction des groupes SCLM d’un projet

Cette fonction extrait les groupes SCLM d’un projet sélectionné.

>>—PROJGRPS——PROJECT—><

Paramètres requis :

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

PROJINFO – Extraction des informations sur un projet SCLM

Cette fonction extrait les informations sur un projet SCLM sélectionné.

>>—PROJINFO——PROJECT—REPDGRP—><

Paramètres requis :

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

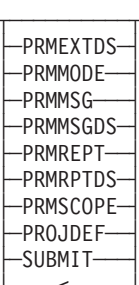
Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

PROMOTE – Promotion d’un membre SCLM

Cette fonction transfère un membre SCLM (ou Archdef) à un niveau supérieur de la hiérarchie des groupes SCLM en fonction de la définition de l'architecture des projets et de la définition du projet.

>>—PROMOTE——GROUP—GROUPPRM—MEMBER—PROJECT—REPDGRP—TYPE—><

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

GROUPPRM

Groupe SCLM - Groupe SCLM à partir duquel le membre requis doit être transféré.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PRMEXTDS [dsn | NONE]]

Fichier d'exit de promotion - NONE ou nom du fichier devant contenir la sortie de l'exit de promotion, si un exit de promotion est utilisé. Si le fichier n'existe pas, un nouveau fichier est alloué. La valeur par défaut est NONE.

[PRMMODE [C | R | U]]

Mode de promotion - Indique le mode de promotion (C=conditionnel, R=rapport, U=inconditionnel). La valeur par défaut est C.

[PRMMSG [Y | N]]

Messages de promotion - L'option Y inclut des messages de promotion. La valeur par défaut est N.

[PRMMSGDS [dsn | NONE]]

Fichier des messages de promotion - NONE ou nom du fichier devant contenir les messages de promotion. Si le fichier n'existe pas, un nouveau fichier est alloué. Les messages de promotion sont également renvoyés dans le fichier de réponses XML, si PRMMSG a la valeur Y. La valeur par défaut est NONE.

[PRMREPT [Y | N]]

Rapport de promotion - L'option Y inclut un rapport de promotion. La valeur par défaut est N.

[PRMRPTDS [dsn | NONE]]

Fichier du rapport de promotion - NONE ou nom du fichier devant contenir le rapport de promotion. Si le fichier n'existe pas, un nouveau fichier est alloué. Les messages de promotion sont également renvoyés dans le fichier de réponses XML, si PRMMSG a la valeur Y. La valeur par défaut est NONE.

[PRMSCOPE [E | N | S]]

Portée de la promotion - Indique la portée de la promotion (E=étendue, N=normale, S=sous-unité). La valeur par défaut est N.

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[SUBMIT [BATCH | ONLINE]]

Méthode de soumission - La promotion est soumise en ligne ou dans un traitement par lots. La valeur par défaut est ONLINE.

REPORT – Création d'un rapport de projet

La fonction de rapport offre un rapport hiérarchique DBUTIL du projet, en fonction des filtres et paramètres de rapport utilisés.



Paramètres requis :

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

REPGRP {group | group* | * }

Groupe SCLM - Groupe(s) SCLM cible du filtrage.

REPMEM {member | mem* | * }

Membre SCLM - Membre(s) SCLM cible du filtrage.

REPTYPE {type | type* | * }

Type SCLM - Type(s) SCLM cible du filtrage.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[REPACODE]

Code des droits d'accès - Code des droits d'accès cible du filtrage.

[REPBMAP [Y | N]]

Mappes de création - Option permettant d'inclure des mappes de création dans le rapport DBUTIL. La valeur par défaut est N.

[REPCCODE]

Code de modification - code de modification cible du filtrage.

[REPLANG]

Langage - Type de langage SCLM cible du filtrage.

[REPMODE [D | E]]

Mode - Mode développeur (D) ou explorateur (E) cible du rapport. La valeur par défaut est D.

REPUTIL –Rapport SCLM DBUTIL

Le service DBUTIL extrait les informations de la base de données de projet et crée un rapport et une sortie personnalisés. Il décrit le contenu de la base de données de projet en fonction des critères de sélection que vous fournissez.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

SAVE – Sauvegarde d'un membre SCLM

Cette fonction copie un membre du répertoire WORKAREA/userid/EDIT/ dans SCLM sous le groupe de développement d'un utilisateur. Un nouveau membre est créé si le membre n'existe pas dans la hiérarchie du projet SCLM.

>>—SAVE——GROUP—MEMBER—PROJECT—TYPE—<<
 └─PROJDEF─┘

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

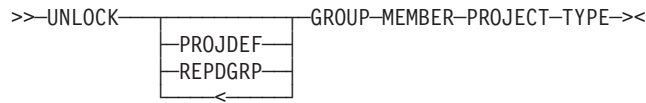
Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

UNLOCK – Déverrouillage d'un membre SCLM

Cette fonction déverrouille un membre SCLM qui était verrouillé par la fonction d'édition (EDIT).



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

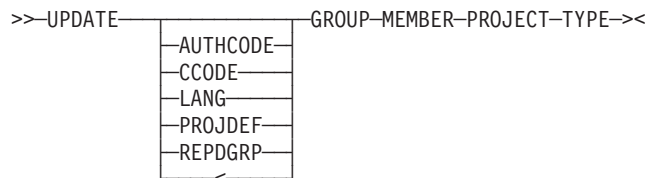
Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[REPDGRP]

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

UPDATE – Mise à jour des informations d'un membre SCLM

Cette fonction met à jour les informations d'un membre dans SCLM.



Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[AUTHCODE]

Code d'autorisation - Code d'autorisation à l'aide duquel le membre est mis à jour.

[CCODE]

Code de modification - Code de modification à l'aide duquel le membre est mis à jour.

[LANG]

Langage SCLM - Langage SCLM du membre sélectionné.

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

[REPDGRP]

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

VERBROW – Versions d'exploration de SCLM

Cette fonction explore une version d'un membre à partir du fichier de version.

```
>>-VERBROW-                    -GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
                    └─PROJDEF─┘
```

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

VERDEL – Suppression de versions SCLM

Cette fonction supprime les informations sur un membre versionné ou audité de SCLM.

```
>>-VERDEL-                    -GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
                    └─PROJDEF─┘
```

Cette fonction supprime toutes les anciennes versions d'un membre dans SCLM.

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

VERHIST – Historique des versions SCLM

Cette fonction affiche l'historique des versions d'une version de membre sélectionnée dans SCLM.

>>-VERHIST-
└──┐ GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
└── PROJDEF ─┘

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

VERLIST – Versions des listes SCLM

Cette fonction répertorie les diverses versions d'un membre particulier.

>>-VERLIST-
└──┐ GROUP-MEMBER-PROJECT-REPDGRP-TYPE-><
└── PROJDEF ─┘

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

VERREC – Versions de restauration SCLM

Cette fonction restaure une version sélectionnée d'un membre dans le groupe de développement.

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

Paramètres requis :

GROUP

Groupe SCLM - Groupe SCLM dans lequel se trouve le membre sélectionné.

MEMBER

Membre SCLM - Membre SCLM sélectionné.

PROJECT

Nom du projet SCLM - Nom du projet SCLM.

REPDGRP

Groupe de développement SCLM - Groupe de développement de l'utilisateur.

TYPE Type SCLM - Type SCLM contenant le membre sélectionné.

Paramètres facultatifs :

[PROJDEF]

Nom de projet SCLM secondaire - Nom de la définition de projet (par défaut, Project).

Exemples

Un exemple de programme Java (avec entrée et sortie) est fourni, qui permet d'accéder à l'API des services hôte de SCLMDT à l'aide d'un serveur HTTP comme mécanisme de transport.

sclmdt_request.xml

L'exemple de demande de l'exemple ci-après appelle la fonction BUILD pour compiler et associer le membre ALLOCEXT dans le projet 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>
```

Figure 24. sclmdt_request.xml – Exemple de fichier de commande XML en entrée

xmlbld.java

L'exemple ci-après illustre un exemple de programme Java (dans Eclipse) utilisant la demande d'entrée XML ci-avant. Lorsqu'il est exécuté, il établit une connexion URL au serveur HTTP sous z/OS (CDFMVS08) et transmet le flux XML en entrée sous forme de demande POST à la routine 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());
        // Below is a sample inline XML input for an ISPF service request
        // This could alternatively be read from an external file

        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>SCLMVC</project>" );
        out.println( "<projdef>SCLMVC</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>" );
        out.println( "</SERVICE-REQUEST>" );
        out.println( "</SCLMDT-INPUT>" );
    }
}

```

Remarques :

1. Dépend de Base64Encoder.class pour le chiffrement des ID utilisateur.
2. Le fichier C:\logon.txt contient USERID:PASSWORD.

sclmdt_response.xml

L'exemple ci-après illustre l'exemple de sortie de la fonction BUILD appelée à l'aide de l'exemple de programme Java.

La figure 26 illustre l'exemple de sortie de la fonction BUILD appelée à l'aide de l'exemple de programme Java.

Figure 26. sclmdt_response.xml – Exemple de fichier de sortie XML

```
START Transfer DATE/TIME is :Wed Mar 26 09:44:03 WST 2008
At url openConnection..
About to accept response from Server
Response from Server received
<?xml version="1.0"?>
<SCLMDT-OUTPUT>
<SERVICE-REQUEST>
<service>SCLM</service>
<session>NONE</session>
<ispprof>IBMUSER.TEST.ISPPROF</ispprof>
<sclmfunc>BUILD</sclmfunc>
<project>SCLMVCM</project>
<projdef>SCLMVCM</projdef>
<member>ALLOCEXT</member>
<group>TEST</group>
<type>SOURCE</type>
<repdgrp>DEV1</repdgrp>
<groupbld>DEV1</groupbld>
<bldrept>Y</bldrept>
<bldmsg>Y</bldmsg>
<bldmode>C</bldmode>
</SERVICE-REQUEST>
<SERVICE-RESPONSE>
  <RETURN-CODE>0</RETURN-CODE>
  <REASON-CODES>
    <REASON-CODE ID="BWB00158">SELECT DETAILS-- BUTTON FOR BUILD MESSAGES,
    REPORTS, LISTINGS</REASON-CODE>
  </REASON-CODES>
  <BUILD-MESSAGES>
    <BUILD-MESSAGE ID="FLM42000">- BUILD PROCESSOR INITIATED - 09:53:01 ON
    2008/03/26</BUILD-MESSAGE>
    <BUILD-MESSAGE ID="FLM46000">- BUILD PROCESSOR COMPLETED - 09:53:01 ON
    2008/03/26</BUILD-MESSAGE>
  </BUILD-MESSAGES>
  <BUILD-REPORT>
    <![CDATA[
      *****
      *****
      **
      **
      **          SOFTWARE CONFIGURATION AND LIBRARY MANAGER (SCLM)          **
      **
      **          B U I L D    R E P O R T          **
      **
      **
    ]]>
  </BUILD-REPORT>
</SERVICE-RESPONSE>
</SCLMDT-OUTPUT>
</xml>
```

```

**                                2008/03/26   09:53:01                                **
**                                                                **
**                                PROJECT:    SCLMVCM                                **
**                                GROUP:      DEV1                                **
**                                TYPE:       SOURCE                                **
**                                MEMBER:     ALLOCEXT                             **
**                                ALTERNATE:  SCLMVCM                             **
**                                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/sclmdt
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 CDFMVS08
Timecheck1:09:52:57.49
Connection Protocol : HTTP
Server Name : CDFMVS08
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/sclmdt
_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=ALLOCEXT&GROUP=TEST
&TYPE=SOURCE&REPDGRP=DEV1&GROUPBLD=DEV1&BLDREPT=Y&BLMSG=Y&BLDMODE=C
/etc/rdz/sclmdt;/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 = ALLOCEXT

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 ALLOCEXT /) 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 ALLOCEXT /)

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_TRANSTABLE=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 ISPZTSO
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 ISPZTSO
Parameters passed to ISPZTSO - PROFILE
Return code from ISPZTSO 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 VCMTEMP
        Parameter to be passed to ISPZTSO 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 ISPZTSO
    Parameters passed to ISPZTSO - 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 ISPZTSO 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: isplib=isp.sisppenu
    Allocation successful for ISPLIB
    REC: ispslib=bzz.sbzsenu,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/sclmdt
    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 PARS *****

```

```

projfile = /etc/rdz/sclmdt/CONFIG/PROJECT/SCLMVCM.conf
Security Flag set to N
rc from FLMSGSG alloc is 0
PARMS=SCLMVCM,SCLMVCM,DEV1,SOURCE,ALLOCEXT,,N,C,Y,Y,,tmpmsg,
    tmpcpt,tmp1st,BLDEXIT
BWBBLD CHECK: PARMS = SCLMVCM,SCLMVCM,DEV1,SOURCE,ALLOCEXT,,
    N,C,Y,Y,,tmpmsg,tmpcpt,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
</ISPFINFO>
ISPRC = 0
***** ISPLG 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 ISPLG 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

```

Annexe D. Utilitaire de génération Rational Application Developer for WebSphere Software

Présentation de l'utilitaire de génération Rational Application Developer for WebSphere

Cette section aborde les améliorations apportées à la procédure de génération Java/J2EE à l'aide de l'utilitaire de génération Rational Application Developer for WebSphere Software (précédemment appelé AST, Application Server Toolkit). L'utilitaire de génération Rational Application Developer for WebSphere Software est fourni avec Rational Application Developer (RAD). Le composant RAD doit être commandé, installé et configuré séparément. L'installation et la personnalisation de ce produit ne sont pas décrites dans le présent ouvrage. Pour connaître les instructions d'installation et de personnalisation de l'utilitaire de génération, reportez-vous à la documentation fournie avec RAD. Cette section fera ultérieurement référence à l'utilitaire de génération Rational Application Developer for WebSphere sous le nom "utilitaire de génération".

L'utilitaire de génération permet à SCLM de générer des espaces de travail de projet dupliqué (qui ont été conservés dans SCLM) à partir de l'environnement de développement intégré RDz à l'aide du mode Eclipse sans interface graphique sous z/OS.

Cette section doit être utilisée avec le Guide d'utilisation en ligne de SCLM Developer Toolkit et le Guide d'installation et de personnalisation fourni avec le produit. SCLM Developer Toolkit fournit des traducteurs de langage Java/J2EE à SCLM afin d'offrir une fonction de génération Java/J2EE complète.

Non seulement SCLM Developer Toolkit offre cette fonction dans le but d'enregistrer du code source et des objets JAVA/J2EE, mais permet également à SCLM, grâce à ces traducteurs, de générer et de contrôler entièrement les applications Java/J2EE. Les objets Java/J2EE pris en charge par le biais de ces traducteurs de langage sont les fichiers classe Java, les fichiers d'archives Java (JAR), les fichiers jar EJB (Enterprise Java Beans), les fichiers d'archive Web (WAR) et les fichiers d'archive d'entreprise (EAR).

SCLM Developer Toolkit s'intègre à l'utilitaire de génération pour contrôler les projets SCLM J2EE répliqués dans l'espace de travail de l'utilitaire de génération sous UNIX System Services sous z/OS. Les projets répliqués sont ensuite générés par le traitement de génération ARCHDEF qui fait référence aux scripts de l'utilitaire de génération. Des exemples de script de génération sont fournis avec SCLM Developer Toolkit. Les objets résultant de la génération/compilation sont stockés dans SCLM.

L'utilitaire de génération prend en charge les générations de tous les projets J2EE acceptés sous Rational Application Developer V7. Les projets générés dans les versions précédentes de RAD devront être migrés dans un espace de travail RAD V7 sur le client IDE avant d'être ajoutés à SCLM. Les projets Java Eclipse normaux sont également pris en charge.

Stockage d'objets Java/J2EE dans SCLM

Les objets Java/J2EE sont stockés dans SCLM de la manière suivante :

- Le source Java est compilé dans des classes. Les classes sont stockées dans SCLM sous le type JAVACLAS. Le nom abrégé et le nom long sont stockés dans les tables de conversion.
- La prise en charge des fichiers EJB et JAR standard (contient les classes et éventuellement d'autres composants de projet Java, comme XML/HTML/JSP dans une structure de module). Les fichiers JAR enregistrés dans le type SCLM J2EEJAR.
- La prise en charge des fichiers WAR assemblés d'après le fichier web.xml J2EE dans le projet J2EE. Les fichiers WAR enregistrés dans le type SCLM J2EEWAR.
- La prise en charge des fichiers EAR créés pour le déploiement d'après le fichier application.xml dans le projet J2EE. Les fichiers EAR enregistrés dans le type SCLM J2EEEAR.
- La prise en charge du déploiement des fichiers EAR dans Websphere Application Server (WAS) sous z/OS.
- La prise en charge de la génération SQLJ
- Toutes les listes en sortie sont enregistrées dans le type SCLM J2EELIST.

Utilitaire de génération comparé à la procédure de génération ANT native

SCLM Developer Toolkit est livré avec une procédure de génération ANT native pour la prise en charge de Java/J2EE. Reportez-vous au plug-in du Guide d'utilisation en ligne de SCLM Developer Toolkit et au Guide d'installation/de personnalisation.

L'utilitaire de génération Rational Application Developer for WebSphere Software améliore la prise en charge de la génération Java/J2EE en répliquant intégralement l'environnement IDE Eclipse sous z/OS lors de la génération.

Comme dans la procédure de génération ANT, la procédure de l'utilitaire de génération utilise la définition d'archive, qui contient les membres composant le projet Java/J2EE, et qui correspond à une représentation abrégée de la manière dont le projet existe dans un espace de travail Eclipse.

L'utilisateur peut choisir la procédure de l'utilitaire de génération ou la procédure de génération ANT native en affectant le script de génération (référéncé par ARCHDEF) avec le convertisseur de langage approprié :

- J2EEAST pour l'utilitaire de génération
- J2EEANT pour la génération ANT native

Lorsqu'elle est générée, la définition d'archive appelle le traducteur de langage de vérification de pré-génération (J2EEAST). Ce type de langage J2EEAST est affecté au script de génération référencé dans la définition d'archive par le mot clé SINC. Ce traducteur de langage copie tous les composants de définition d'archive dans le système de fichiers z/OS UNIX System Services à générer à l'aide de l'utilitaire de génération.

Les composants de projet et la source Java peuvent être stockés dans SCLM au format EBCDIC ou ASCII. Sous l'utilitaire de génération, les composants de texte et la source Java sont convertis par défaut au format ASCII dans l'espace de travail USS avant le traitement de génération.

Remarques sur l'installation de l'utilitaire de génération Rational Application Developer for WebSphere Software

L'utilitaire de génération Rational Application Developer for WebSphere Software est fourni avec Rational Application Developer (RAD). Le composant RAD doit être commandé, installé et configuré séparément. L'installation et la personnalisation de ce produit ne sont pas décrites dans le présent ouvrage. Pour connaître les instructions d'installation et de personnalisation de l'utilitaire de génération, reportez-vous à la documentation fournie avec RAD. Le module du répertoire de l'utilitaire de génération doit se trouver dans le système de fichiers z/OS UNIX.

Le traitement de génération J2EE peut nécessiter des tailles de région considérables pour éviter les erreurs de stockage ou de 'mémoire insuffisante'. En prévision de générations en ligne volumineuses, spécifiez au moins REGION=512M sur le démon Developer for System z RSE (par défaut, il s'agit de la tâche démarrée RSED). Pour le traitement par lots, il est recommandé que ce paramètre de taille de région soit spécifié sur le lot JOBCARD.

Intégration de SCLM à l'utilitaire de génération

Le traitement J2EEAST initial est similaire au traitement J2EEANT dans lequel le traducteur de langage détermine à partir de la définition d'archive quels composants doivent être régénérés en fonction du mode de génération (conditionnel ou forcé). Les fichiers source des projets et les objets générés inclus sont ensuite copiés dans la zone de travail du système de fichiers UNIX Systems Services pour l'utilitaire de génération. Le script d'exécution et le fichier XML de génération stockés dans SCLM sous le type J2EEBLD sont également copiés dans la même zone de travail. Le script d'exécution est appelé pour afficher Eclipse sans interface graphique sous z/OS et exécuter le code XML de génération de l'application référencée. L'utilitaire de génération compile et génère les objets Java/J2EE requis spécifiés par le script d'exécution, les données XML de génération et la définition d'archive.

J2EEAST examine les objets générés et ne sélectionne pour la mise à jour de SCLM que les composants/objets qui devaient être régénérés selon SCLM à partir du mode de génération (conditionnel/forcé).

SCLM traite chaque composant de définition d'archive en exécutant chaque traducteur de langage associé au composant. Le traducteur de langage JAVA associé à chaque membre de source Java copie les fichiers de classe associés à chacun dans SCLM.

La partie finale du traitement de génération consiste à traiter la définition d'archive (ARCHDEF) elle-même, où le traducteur de langage J2EEOBJ est exécuté. Ce traducteur de définition d'archive détermine quels objets J2EE ont été générés (JAR, WAR, EAR) et les recopie dans SCLM.

Implémentation et exécution de l'utilitaire de génération Rational Application Developer for WebSphere Software

L'implémentation et l'exécution de l'utilitaire se déroulent de la manière suivante :

- Par défaut, SCLM Developer Toolkit est fourni avec des assistants de script de génération et de définition d'archive qui génèrent les définitions d'archive des projets et les scripts de génération associés. Ces assistants génèrent des scripts de procédure de génération ANT natifs et affectent les scripts de génération avec le

langage J2EEANT. La page des préférences de génération SCLM de SCLM Developer Toolkit contient une case à cocher permettant à ces assistants de générer à la place des scripts de génération.

- Dans la page **Equipe > Préférences du SCLM > Options de script de génération**, chaque exemple de script de génération (Java, EJB, WAR, EAR, SQLJ) doit être associé à un script d'exécution BWBASTR.
- Le script d'exécution (exemple de BWBASTR) doit être copié dans chaque projet SCLM qui requiert l'utilitaire de génération. Le script doit se trouver dans le type SCLM J2EEBLD et posséder le type de langage TEXT ou J2EEPART. Ce script doit être personnalisé. Les instructions de personnalisation sont fournies dans le script d'exécution lui-même. Pour plus d'informations sur le script BWBASTR, reportez-vous aux formats et aux scripts de l'utilitaire de génération.
- Chaque composant de projet (JAR, WAR, EAR) requiert une définition d'archive (reportez-vous aux scripts de l'utilitaire de génération pour la disposition du format de la définition d'archive) et un script de génération correspondant (reportez-vous à la section 2.4 pour la disposition des scripts de génération). Ces scripts de génération doivent être créés dans un type SCLM J2EEBLD et être définis avec un type de langage J2EEAST. Ces scripts de génération et définitions d'archive peuvent être créés par l'utilisateur dans SCLM ou l'utilisateur peut les générer à l'aide des assistants de script de génération et de définition d'archive du client, en sélectionnant **EQUIPE > Générer un script de génération Java/J2EE** ou **EQUIPE > Ajouter à SCLM**.
- Les détails relatifs à l'incident de génération des composants ou de compilation seront renvoyés dans le journal des opérations de génération. Si une génération échoue, vous pouvez repérer les erreurs de compilation en recherchant la "LISTE DE COMPILATION" dans le journal des opérations. Si le journal indique une erreur relative à la configuration ou l'installation de l'utilitaire de génération, signalez-la à l'administrateur système. Les journaux des incidents de génération AST Eclipse se trouvent dans le système de fichiers UNIX System Services, sous le répertoire AST_INSTALL/Eclipse/configuration. Ces journaux d'erreurs spécifiques sont illisibles sous z/OS car ils sont au format ASCII. Ils doivent être convertis en EBCDIC ou transférés en mode binaire vers le bureau Windows® pour pouvoir être consultés.
- Si vous exécutez des générations J2EE par lots, il est recommandé d'inclure le paramètre de taille de région REGION=512M dans la carte des travaux par lots. Si vous spécifiez une taille de région plus petite, des incidents de stockage 'mémoire insuffisante' peuvent se produire.

Traducteurs de langage AST de SCLM

Pour l'utilitaire de génération, certains traducteurs de langage sont affectés au source J2EE, à ARCHDEF et au script de génération.

Des exemples de génération de ces langages dans un projet SCLM sont fournis avec SCLM Developer Toolkit dans la bibliothèque des exemples d'installation, SBWBSAMP.

Source Java

Langage = JAVA | JAVABIN

Texte J2EE

Langage = J2EEPART | J2EEBIN (par exemple, XML)

Binaire J2EE

Langage = J2EEBIN (par exemple, jpg)

script de génération

Langage = J2EEAST

script d'exécution

Langage = TEXT | J2EEPART

ARCHDEF

Langage = archdef

Remarque : Le mot clé LKED=J2EEOBJ traite la définition d'archive avec le traducteur de langage J2EEOBJ.

J2EEAST : Traducteur de génération/vérification J2EE

Il s'agit d'un traducteur de langage du code XML du script de l'utilitaire de génération.

Récapitulatif : EXEMPLE : BWBTRAN4

Le traducteur de vérification J2EEAST est appelé lors de la génération de la définition d'archive. J2EEAST est le type de langage du code XML de génération J2EEBLD référencé par le mot clé SINC dans la définition d'archive. Ce traducteur de vérification copie la source sélectionnée et tous les objets de définition d'archive, quel que soit le mode de génération dans la zone de travail HFS z/OS à référencer lors de la génération comme espace de travail Projet. Cet espace de travail de projets se trouve provisoirement dans la zone de travail de SCLM Developer Toolkit. SCLM DT crée un espace de travail de projets temporaire sous l'arborescence de répertoires suivante : /var/SCLMDT/WORKAREA/userid/project/group/type/member/project

L'utilitaire de génération requiert l'intégralité de la structure du projet et tous les composants de projet sont donc copiés dans le système de fichiers UNIX System Services. Tous les composants texte sont copiés dans l'espace de travail des projets au format ASCII. Le code XML de génération est également copié dans la zone de travail du système hiérarchique de fichiers. Le code XML de génération contient les détails du projet et les demandes de génération J2EE, ainsi que les détails des autres propriétés référencés par SCLM.

Le code XML de génération fait référence à un script d'exécution AST associé. Ce script d'exécution est également copié dans la zone de travail (EBCDIC) et, lorsqu'il est exécuté, affiche Eclipse sans interface graphique sous z/OS pour exécuter les données XML de génération de l'application. L'implémentation actuelle permet une génération intégrale dans l'espace de travail mais SCLM ne traite les composants sélectionnés que si une génération conditionnelle a été demandée et que le traducteur détermine quels objets générés doivent être mis à jour dans SCLM, selon le mode de génération.

Les objets J2EE générés, tels que les fichiers d'archive JAR, WAR ou EAR seront de nouveau stockés dans SCLM lorsque le traducteur de définition d'archive J2EEOBJ est exécuté. Les fichiers de classe sélectionnés sont consignés et mis à jour dans SCLM lorsque la définition d'archive traite chaque composant Java (traducteur Java).

J2EEOBJ : Traducteur ARCHDEF J2EE

Il s'agit d'un traducteur de langage de définition d'archive référencé par le mot clé LKED.

Récapitulatif : EXEMPLE : BWBTRAN3

Ce traducteur de génération final est appelé dans le cadre du processus de génération de la définition d'archive. Ce traducteur détermine quels objets J2EE étaient déjà générés dans le traducteur J2EEAST et copie ces objets dans SCLM avec le nom abrégé indiqué.

1. Si la variable SCLM_BLDMAP est définie, extrayez les informations sur la mappe de génération et mettez à jour l'objet J2EE dans le répertoire META-INF.
2. Copiez ces objets J2EE (JAR, WAR, EAR) dans SCLM avec le nom abrégé associé.
 - JAR/EJB dans le type SCLM J2EEJAR
 - WAR dans le type SCLM J2EEWAR
 - EAR dans le type SCLM J2EEEAR
3. Copiez les fichiers de transactions du journal de génération dans le type SCLM J2EELIST

JAVA : Traducteur de langage Java (EBCDIC)

Il s'agit d'un traducteur de langage de source Java stocké en EBCDIC.

Récapitulatif : EXEMPLE : BWBTRAN1

Le type de langage pour source Java est défini par l'exemple BWBTRAN1. Le traducteur Java détermine le type de génération effectué sur la source Java. Remarque : Cette définition de langage doit être affectée aux programmes Java si vous souhaitez enregistrer le code source Java en EBCDIC sur l'hôte (à savoir que le source peut être affiché et modifié directement sur l'hôte via ISPF). L'avantage de définir des programmes avec cette définition de langage est de pouvoir modifier et afficher la source directement sur l'hôte z/OS. Par contre, les pages de codes doivent être converties lors de la migration ou de l'importation de projets du client vers l'hôte.

JAVABIN : Traducteur de langage Java (ASCII)

Il s'agit d'un traducteur de langage de source Java stocké en ASCII.

Récapitulatif : EXEMPLE : BWBTRAN1

Type de langage similaire à Java et utilisé lors du stockage de la source Java au format ASCII dans SCLM.

J2EEPART : Traducteur de langage texte J2EE

Il s'agit d'un traducteur de langage de composant texte stocké en EBCDIC.

Récapitulatif : EXEMPLE : BWBTRANJ

J2EEPART est un type de langage qui spécifie un composant JAVA/J2EE et qui est défini par l'exemple BWBTRANJ. Aucune analyse syntaxique particulière n'est effectuée lors de la génération de cette définition de langage. Les sources autres que Java ou les composants J2EE qui requièrent une conversion de langage ASCII/EBCDIC peuvent être attribués de manière générique sous cette définition de langage si aucune analyse de génération particulière n'est requise (par exemples, HTML, XML, .classpath, .project, tables de définition). La définition de langage TEXT peut éventuellement être utilisée.

J2EEBIN : Traducteur de langage binaire J2EE

Il s'agit d'un traducteur de langage de source Java stocké en EBCDIC.

Récapitulatif : EXEMPLE : BWBTRANJ

Type de langage qui spécifie un composant ASCII ou binaire JAVA/J2EE et qui est défini par l'exemple BWBTRANJ. Aucune analyse syntaxique particulière n'est effectuée lors de la génération de cette définition de langage. Les fichiers binaires JAVA/J2EE et les fichiers texte que vous souhaitez enregistrer au format ASCII peuvent être attribués de manière générique sous cette définition de langage si aucune analyse de génération particulière n'est requise.

Formats et scripts de génération de l'utilitaire de génération

Comme pour la méthode du service de génération J2EE standard, la méthode de l'utilitaire de génération implique que les scripts de l'utilitaire de génération soient référencés par la définition d'archive (ARCHDEF) à l'aide du mot clé SINC.

Le script de génération appelé est un XML de génération d'application de l'utilitaire de génération qui doit être personnalisé et unique pour chaque projet J2EE. Ce script de génération fait également référence à un script d'exécution de l'utilitaire de génération qui contient les propriétés globales de cet utilitaire et les commandes d'exécution Eclipse permettant d'afficher Eclipse sous z/OS sans interface graphique et d'exécuter le XML de génération sélectionné. Généralement, le script BWBASTR personnalisé est utilisé par tous les scripts de génération Build. Le type de langage de tous les scripts de génération de l'utilitaire de génération doit être J2EEAST. Le type de langage du script BWBASTR doit être un traducteur de langage en texte seul, tel que TEXT ou J2EEPART.

Le script d'exécution de l'utilitaire de génération (BWBASTR) et les exemples de script de génération par projet (projet Java/Jar (BWBASTJ), projet EJB (BWBASTEJ), projet de client d'application (BWBASTAP), projet Web (BWBASTW) et projet EAR (BWBASTE)) peuvent être copiés de la bibliothèque d'exemples de l'utilitaire de génération de SCLM Developer Toolkit dans les projets SCLM des clients sous le type J2EEBLD.

Format du script de génération

Le format du script de génération inclut les éléments suivants :

SCLM_ARCHDEF

Nom de la définition d'archive référencée générée

AST_SHSCRIPT

Nom du script d'exécution AST permettant d'exécuter AST sous z/OS.
(voir l'exemple BWBASTR)

PROJECT NAME

Nom du projet J2EE (il ne s'agit pas du nom de projet SCLM)

JAR_FILE_NAME

Pour un projet Java, nom du fichier JAR généré

WAR_NAME

Pour un projet Web, nom du fichier WAR généré

EJB_NAME

Pour un projet EJB, nom du fichier JAR généré

EAR_NAME

Pour une application d'entreprise, nom du fichier EAR généré qui peut être déployé

SCLM_BLDMAP

Si la valeur est YES, inclure dans le répertoire MANIFEST du fichier JAR, WAR et EAR. Fournit l'audit et la mappe de génération des composants inclus.

Les routines ANT d'AST sont également incluses pour générer l'objet requis. Les scripts de génération SCLM requis sont des versions modifiées des exemples ci-après fournis par AST :

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

Routines AST non modifiées

Les routines ci-après ont été extraites du guide Application Server Toolkit.

projectImport : Cette tâche importe un système de fichiers existant dans un espace de travail.

Tableau 13. Paramètres projectImport

Attribut	Description	Obligatoire
ProjectName	Nom du projet à importer	Oui
ProjectLocation	Emplacement complet du projet (sous l'espace de travail ou ailleurs dans le système de fichiers).	Non, la valeur par défaut est \${workspaceLocation}/\${projectName}

Exemple : Importez un projet sous le répertoire de l'espace de travail, mais qui ne se trouve pas dans l'espace de travail :

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

projectBuild : Cette tâche génère le projet spécifié.

Tableau 14. Paramètres projectBuild

Attribut	Description	Obligatoire
ProjectName	Nom du projet à générer	Oui
BuildType	Type de génération	Non, la valeur par défaut est Par incrément. Valeurs possibles : Par incrément et Saturé.
FailOnError	Indique si les générations doivent échouer en cas d'erreur	Non, la valeur par défaut est true.
DebugCompilation	Indique si les compilations doivent être déboguées	Non, la valeur par défaut est true.
Concis (déprécié)	Indique si les messages doivent être imprimés	Non, la valeur par défaut est false.
ShowErrors	Indique si le journal des générations Ant doit contenir les erreurs de projet	Non, la valeur par défaut est true.
SeverityLevel	Niveau d'incident à comptabiliser et traiter comme une erreur de génération	Non, la valeur par défaut est ERROR. Valeurs possibles : ERROR, WARNING ou INFORMATION.
CountValidationErrors	Indique si les incidents de validation doivent être comptabilisés comme des erreurs de projet	Non, la valeur par défaut est true.

Tableau 14. Paramètres projectBuild (suite)

Attribut	Description	Obligatoire
PropertyCountName	Propriété permettant de recevoir le nombre d'erreurs de projet	Non, la valeur par défaut est ProjectErrorCount.
PropertyMessagesName	Propriété permettant de recevoir les messages d'erreurs du projet	Non, la valeur par défaut est ProjectErrorMessages.

Exemples :

- Générez "myProject". Il s'agit par défaut d'une génération par incrément avec des informations de débogage :

```
<projectBuild ProjectName="myProject" />
```
- Effectuez une génération de production de "myProject", une génération complète sans informations de débogage :

```
<projectBuild
  ProjectName="myProject"
  failonerror="true"
  DebugCompilation="false"
  BuildType="full" />
<echo message="projectBuild: projectName=${projectName}
project Error Count=${ProjectErrorCount}
project Error Messages=${ProjectErrorMessages}" />
```

EJBexport : Cette tâche effectue la même opération que l'assistant d'exportation de fichier JAR EJB pour exporter un projet EJB dans un fichier Jar EJB. Cette tâche n'est pas disponible dans les produits qui n'incluent pas d'outils de développement EJB.

Tableau 15. Paramètres EJBexport

Attribut	Description	Obligatoire
EJBProjectName	Nom du projet EJB (Sensible à la casse)	Oui
EJBExportFile	Chemin d'accès absolu du fichier JAR EJB.	Oui
ExportSource	Indique si les fichiers source doivent être inclus.	Non, la valeur par défaut est false.
Overwrite	Indique si le fichier existant doit être remplacé.	Non, la valeur par défaut est false.

Exemple : Exportez le projet "EJBProject" dans "EJBProject.jar"

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

WARExport : Cette tâche effectue la même opération que l'assistant d'exportation de fichier WAR pour exporter un projet Web dans un fichier WAR.

Tableau 16. Paramètres WARExport

Attribut	Description	Obligatoire
WARProjectName	Nom du projet Web (Sensible à la casse)	Oui
WARExportFile	Chemin d'accès absolu du fichier WAR	Oui
ExportSource	Indique si les fichiers source doivent être inclus.	Non, la valeur par défaut est false.

Tableau 16. Paramètres WARExport (suite)

Attribut	Description	Obligatoire
Overwrite	Indique si le fichier existant doit être remplacé.	Non, la valeur par défaut est false.

Exemple : Exportez le projet "ProjectWeb" dans "ProjectWeb.war"

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

EARExport : Cette tâche effectue la même opération que l'assistant d'exportation de fichier WAR pour exporter un projet Web dans un fichier WAR.

Tableau 17. Paramètres EARExport

Attribut	Description	Obligatoire
EARProjectName	Nom du projet d'application d'entreprise (Sensible à la casse)	Oui
EARExportFile	Chemin d'accès absolu du fichier EAR	Oui
ExportSource	Indique si les fichiers source doivent être inclus.	Non, la valeur par défaut est false.
IncludeProjectMetaFiles	Indique si les fichiers de métadonnées de projet qui contiennent entre autres le chemin de génération Java et les noms de projet doivent être inclus. Utilisé lors de la réimportation comme projets binaires.	Non, la valeur par défaut est false.
Overwrite	Indique si le fichier existant doit être remplacé.	Non, la valeur par défaut est false.

Exemple : Exportez le projet "EARProject" dans "EARProject.ear"

```
<earExport EARProjectName="EARProject" EARExportFile="EARProject.ear"/>
```

AppClientExport : Cette tâche effectue la même opération que l'assistant d'exportation de fichier WAR pour exporter un projet Web dans un fichier WAR.

Tableau 18. Paramètres AppClientExport

Attribut	Description	Obligatoire
AppClientProjectName	Nom du projet de client d'application (Sensible à la casse)	Oui
AppClientExportFile	Chemin d'accès absolu du fichier JAR du client d'application	Oui
ExportSource	Indique si les fichiers source doivent être inclus.	Non, la valeur par défaut est false.
Overwrite	Indique si le fichier existant doit être remplacé.	Non, la valeur par défaut est false.

Exemple : Exportez le projet "ProjectClient" dans "ProjectClient.jar" :

```
<appClientExport
AppClientProjectName="ProjectClient"
AppClientExportFile="ProjectClient.jar"/>
```

Script d'exécution AST (exemple BWBASTR) : Le script d'exécution AST est référencé par les scripts de génération AST (type=J2EEBLD lang=TEXT).

```

/*

Cet exemple de script de génération permet de générer des projets J2EE à l'aide
d'AST (Application Server toolkit) sous z/OS.
AST est un générateur Eclipse sans interface graphique, installable
de manière distincte en dehors de SCLM Developer toolkit.
Cet exemple de script doit être personnalisé et se trouver dans le type SCLM

du J2EEBLD
appelé à partir d'un membre de définition d'archive J2EE via le mot clé de définition
d'archive SINC. Il doit être stocké avec un langage de texte tel que TEXT ou J2EEPART.

Les arguments BUILDFILE et WORKSPACE sont transmis en interne à partir du
traducteur de langage AST J2EE de SCLM Developer Toolkit

*/

parse arg $args
parse var $args BUILDFILE WORKSPACE

/*----- Personnalisation de l'utilisateur -----*/

/* Personnalisez les valeurs de variable suivantes : AST_DIR et JAVA_DIR */

/* AST_DIR est le répertoire Eclipse z/OS où AST est installé
*/
AST_DIR='/u/AST/eclipse'
/* JAVA_DIR est le répertoire bin Java z/OS approprié
*/
/* Spécifiez le bin Java fourni avec le package AST
*/
JAVA_DIR='/u/AST/eclipse/jdk/bin'

/*----- Fin de la personnalisation de l'utilisateur -----*/
/* Il n'est pas nécessaire de modifier le reste de l'exemple */

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
End

/* Les paramètres ci-après sont définis pour un appel d'Eclipse sans interface
graphique */
/* Options de paramètre de mémoire Java comprises entre 256 Mo et 512 Mo définies */
/* Augmentez la valeur du paramètre de mémoire maximale en cas d'erreur de mémoire

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

/* Les options de vidage Java ont été désactivées pour empêcher les vidages Java */
/* lors d'incidents d'allocation de mémoire */

```

```

JAVA_NODUMP='export JAVA_DUMP_OPTS="ONANYSIGNAL(NONE)"'
JAVA_CMD  = JAVA_DIR'/java 'PARGS
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
    End

EXIT 0

```

Script de génération Java/JAR (exemple) : Le script de génération suivant est type=J2EEBLD lang=J2EEAST. Il accepte tout nom de huit caractères ou moins. Les variables sont définies au format XML standard.

```
<!--
```

BWBASTJ : Exemple de JAR J2EE pour AST

Il s'agit d'un exemple de script XML de génération à exécuter à l'aide d'AST Eclipse sans interface graphique sous z/OS.

Le script de génération SCLM est copié dans le répertoire de zone de travail approprié sur le système de fichiers USS z/OS.

Le mot clé projectlocation est remplacé de manière dynamique par l'espace de travail affecté approprié. Vérifiez que la ligne projectlocation est conservée en l'état :

Personnalisez le projet et les variables de propriétés ci-après

```

project name : Remplacez ASTJAR par le nom de projet de l'application
               Java
AST_SHSCRIPT : Nom du squelette du script d'exécution AST pour la génération
SCLM_ARCHDEF : Nom de la définition d'archive à générer
JAR_FILE_NAME : Nom du fichier JAR créé
SCLM_BLDMAP  : Si la valeur est YES, inclure dans le répertoire MANIFEST

```

du fichier JAR.

Fournit un audit et une mappe de génération pour les composants

```
inclus
```

```
-->
```

```

<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 génération WAR (exemple) : Le script de génération suivant est type=J2EEBLD lang=J2EEAST.

```
<!--
```

BWBASTW : Exemple J2EE pour AST

Il s'agit d'un exemple de script XML de génération à exécuter à l'aide d'AST Eclipse sans interface graphique sous z/OS.

Le script de génération SCLM est copié dans le répertoire de zone de travail approprié sur le système de fichiers USS z/OS.

Le mot clé projectlocation est remplacé de manière dynamique par l'espace de travail affecté approprié. Vérifiez que la ligne projectlocation est conservée en l'état :

Personnalisez le projet et les variables de propriétés ci-après

```

project name   : Remplacez ASTWAR par le nom de projet de l'application
                WAR
AST_SHSCRIPT   : Nom du squelette du script d'exécution AST pour la génération
SCLM_ARCHDEF   : Nom de la définition d'archive à générer
WAR_NAME       : Nom du fichier WAR créé
SCLM_BLDMAP    : Si la valeur est YES, inclure dans le répertoire MANIFEST du
                fichier WAR.
                Fournit un audit et une mappe de génération pour les composants

```

```
inclus
```

```
-->
```

```

<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 génération EJB (exemple) : Le script de génération suivant est
type=J2EEBLD lang=J2EEAST.

<!--

BWBASTEJ : Exemple EJB J2EE pour AST

Il s'agit d'un exemple de script XML de génération à exécuter à l'aide d'AST Eclipse sans interface graphique sous z/OS.

Le script de génération SCLM est copié dans le répertoire de zone de travail approprié sur le système de fichiers USS z/OS.

Le mot clé projectlocation est remplacé de manière dynamique par l'espace de travail affecté approprié. Vérifiez que la ligne projectlocation est conservée en l'état :

Personnalisez le projet et les variables de propriétés ci-après

```
project name : Remplacez ASTEJB par le nom de projet de l'application
               EJB
AST_SHSCRIPT : Nom du squelette du script d'exécution AST pour la génération
SCLM_ARCHDEF : Nom de la définition d'archive à générer
EJB_NAME      : Nom du fichier JAR EJB créé
SCLM_BLDMAP   : Si la valeur est YES, inclure dans le répertoire MANIFEST du
               fichier JAR, WAR ou EAR.
               Fournit un audit et une mappe de génération pour les composants
```

inclus

-->

```
<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 génération EAR (exemple) : Le script de génération suivant est
type=J2EEBLD lang=J2EEAST.

<!--

BWBASTE : Exemple d'EAR J2EE pour AST

Il s'agit d'un exemple de script XML de génération à exécuter à l'aide d'AST Eclipse sans interface graphique sous z/OS.

Le script de génération SCLM est copié dans le répertoire de zone de travail approprié sur le système de fichiers USS z/OS.

Le mot clé projectlocation est remplacé de manière dynamique par l'espace de travail affecté approprié lors de la génération.

Vérifiez que la ligne projectlocation est conservée en l'état.

Personnalisez le projet et les variables de propriétés ci-après

project name : Remplacez ASTEAR par le nom de projet de l'application
EAR
AST_SHSCRIPT : Nom du squelette du script d'exécution AST pour la génération
SCLM_ARCHDEF : Nom de la définition d'archive à générer
EAR_NAME : Nom du fichier EAR créé
SCLM_BLDMAP : Si la valeur est YES, inclure dans le répertoire MANIFEST du
fichier JAR, WAR ou EAR.
Fournit un audit et une mappe de génération pour les composants

inclus

-->

```
<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>
```

Format d'ARCHDEF J2EE : Le format d'ARCHDEF est le même que pour la procédure de génération J2EEANT normale.

SINC Le source inclut le script de génération J2EEBLD.

INCLD

SCLM inclut le composant J2EE (par exemple, source Java).

INCL SCLM inclut une autre définition d'archive

OUT1 Indique le type d'objet J2EE créé par cette définition d'archive :

- J2EEEAR
- J2EEWAR
- J2EEJAR

LIST Correspond à la liste récapitulative des composants et de l'audit de la génération de l'ARCHDEF. Contenu dans TYPE=J2EELIST sous le nom de membre de la définition d'archive.

LKED

Indique qu'il s'agit d'une définition d'archive LEC et spécifie le langage du traducteur de définition d'archive à appeler (toujours J2EEOBJ pour les définitions d'archive J2EE).

```

SINC  SCRIPT1      J2EEBLD
INCLD XX000001    SOURCE
INCL  PAULWAR2    ARCHDEF

OUT1  *            J2EEEAR
LIST  *            J2EELIST
LKED  J2EEOBJ

```

Prise en charge de la génération SQLJ

SCLM permet la prise en charge de SQLJ via les traducteurs Java/J2EE livrés avec SCLM Developer Toolkit. Ces traducteurs, conjointement avec les scripts de génération AST permettent de stocker et de générer des projets sqlj. Ils offrent des points d'intégration permettant aux routines client de définir des propriétés de personnalisation sqlj db2 et prenant en charge le traitement des liaisons db2 par l'intermédiaire des étapes de génération et de promotion de SCLM via les exits de génération et de promotion de SCLM.

Pour plus d'informations sur la prise en charge de SQLJ, reportez-vous au Guide d'utilisation de SCLM Developer Toolkit.

La prise en charge de SQLJ dans SCLM (à l'aide de la méthode AST) peut se résumer ainsi :

- Stockage du source SQLJ dans SCLM et affectation au source d'un langage de type SQLJ
- Génération d'une définition d'archive Java/J2EE qui peut inclure des membres SQLJ dans la définition d'archive. La définition d'archive fait référence à un script de génération SQLJ de type J2EEBLD qui contiendra les propriétés sqlj à définir et personnaliser par l'utilisateur (voir les exemples de script BWBASTSQ et BWBASTSE).
- Post-traitement facultatif des liaisons DB2 (à l'aide des fichiers de module d'interrogation de base de données générés et stockés dans SCLM).
Post-traitement des liaisons à piloter par les exits utilisateur de génération et de promotion. Fichiers de module d'interrogation de base de données accessibles directement dans les fichiers contrôlés par SCLM.
- Déploiement du JAR/EAR généré via le traitement du déploiement SCLM DT

Personnalisation de SQLJ (pour l'administrateur SCLM)

Configuration requise : Le support sqlj DB2 est installé. Le répertoire db2 ci-après (ou un répertoire similaire selon le répertoire d'installation du site) doit se trouver dans le système de fichiers USS z/OS : /usr/lpp/db2/db2810/* (il s'agit du répertoire de db2 v8). Ce répertoire est requis pour la personnalisation du script de génération SQLJ. Pour SQLJ, des dépendances de chemin d'accès aux classes existent sur /usr/lpp/db2/db2810/jcc/classes/sqlj.zip. Pour db2sqljcustomize, les dépendances de chemin d'accès aux classes existent sur /usr/lpp/db2/db2810/jcc/classes/db2jcc.jar.

Traducteurs de langage :

SQLJ : Un nouveau traducteur de langage SQLJ est fourni et doit être attribué en tant que type de langage à l'ensemble du code source SQLJ enregistré dans SCLM. Ce nouveau traducteur requiert la définition de types SCLM supplémentaires. Le nouveau traducteur est similaire au traducteur JAVA mais il contient des définitions IOTYPE supplémentaires pour les types de sortie SCLM SQLJSER et DBRMLIB. Si le client ne génère pas de fichiers de module d'interrogation de base de données dans le cadre de l'étape db2sqljcustomize, cet élément DBRMLIB

IOTYPE peut être retiré de la définition de langage SQLJ. (Référence à l'exemple de définition de traducteur SQLJ BWBTRANS). Dans le projet PROJDEF, générez le nouveau traducteur SQLJ et les types supplémentaires, comme suit :

- SQLJSER : Type requis pour contenir les fichiers de profil sérialisés générés (fichiers .ser) créés/personnalisés dans les étapes sqlj et db2sqljcustomize. Il est recommandé de définir cet ensemble de données de type SCLM en tant que recfm=VB, lrecl= 256.
- DBRMLIB : Type devant contenir les fichiers de modules d'interrogation de base de données générés créés dans l'étape. Ce type est requis uniquement pour les clients utilisant les fichiers de modules d'interrogation de base de données générés dans le cadre du traitement de liaison DB2. Il est recommandé de définir cet ensemble de données de type SCLM en tant que recfm=VB, lrecl= 256.

Personnalisation de l'utilisateur SQLJ

Script des propriétés de génération SQLJ : Les clients doivent définir les propriétés DB2 SQLJ dans le script de génération SQLJ principal (exemple BWBASTSQ) qui se trouve dans le type J2EEBLD. Ces propriétés seront utilisées dans les étapes "sqlj" et "db2sqljcustomize".

Remarque : Le squelette SQLJ BWBASTSQ fourni n'est qu'une extension du squelette de génération Java normal, BWBASTJ. En outre, BWBASTSE est une extension sqlj de l'exemple d'EJB BWBASTEJ. Les clients peuvent mélanger les membres de source SQLJ et Java dans la même définition d'archive et utiliser BWBASTSQ. La génération résultante crée un fichier JAR contenant tous les fichiers .ser et de classe générés.

Vous trouverez ci-après une liste des définitions de propriétés "sqlj" et "db2sqljcustomize" requises.

Paramètres des propriétés générales

```
<!-- spécifiez l'emplacement d'exécution du bin JAVA -->
<property name="JAVA_BIN" value="/usr/lpp/java/J5.0/bin"/>
```

Sqlj

```
<!-- spécifiez l'emplacement de la routine d'exécution sqlj et

db2sqljcustomize
    bwbsqlc.rex (exemple BWBSQLC) -->
<!-- Par défaut, cet exemple doit se trouver dans le répertoire

d'installation de
    SCLM DT ou y être copié -->
<property name="sqlj.exec" value="/etc/SCLMDT/bwbsqlc.rex"/>

<!-- specify global property arguments below for sqlj processing -->
<property name="sqlj.arg" value="-compile=false -status
    -linemap=NO -db2optimize"/>
```

db2sqljcustomize

```
<!-- spécifiez l'emplacement du fichier db2jcc.jar à inclure dans le chemin
d'accès aux classes db2sqljcustomize -->
<property name="db2sqljcustomize.cp" value="/usr/lpp
    /db2/db2810/jcc/classes/db2jcc.jar":./SRC:/usr/lpp/db2810/jcc/
    classes/db2jcc_license_cisuz.jar"/ >

<!-- spécifiez les arguments des propriétés globales ci-après pour

db2sqljcustomize -->
```

```
<property name="db2sqljcustomize.arg" value='-automaticbind NO -onlinecheck
      YES -staticpositioned YES -bindoptions "ISOLATION(CS)" -genDBRM' />
```

<!-- Vous trouverez ci-après le nom du fichier de propriétés temporaire à transmettre à une routine de propriétés utilisateur pour stocker des propriétés d'argument supplémentaires.
Aucune personnalisation n'est requise -->

```
<property name="db2sqljcustomize.propfile" value="user.properties" />
```

<!-- Vous trouverez ci-après le nom d'un programme utilisateur facultatif qui sera exécuté immédiatement avant le processus db2sqljcustomize.
Il met à jour dynamiquement un fichier de propriétés db2sqljcustomize.

propfile à
utiliser comme entrée de la commande db2sqljcustomize
La routine db2sqljcustomize (property: sqlj.exec) concatènera et utilisera à la fois les propriétés d'argument définies dans ce fichier

build.xml
(db2sqljcustomize.arg) et les propriétés d'argument lues à partir du fichier des propriétés utilisateur.
La routine utilisateur recevra les arguments suivants :
basedir : Répertoire de base correspondant au répertoire de l'espace

de travail
propfile : Nom du fichier de propriétés à créer (fichier temporaire à créer dans l'espace de travail)
sqljf : Tous les noms de fichier .ser à traiter par db2sqljcustomize

Remarque : Le fichier de propriétés doit être basedir/'propfile
Les propriétés doivent être définies dans le fichier au format

suivant
argument=value (eg: singlepkgname=longname:pkgname)
(une déclaration de propriété par ligne)
eg:
singlepkgname=src/TeSQLJ986_SJProfile0.ser:SQLJ986
singlepkgname=src/TeSQLJ987_SJProfile0.ser:SQLJ987
pkgversion=1
url=jdbc:db2://site1.com:80/MVS01
qualifier=DBT

Si aucun programme utilisateur n'est requis, spécifiez :
<property name="sqlj.userpgm" value="NONE" />

-->

```
<property name="sqlj.userpgm" value="/u/userdir/BWBSQLD" />
```

*** fin des paramètres de propriété ***

Les propriétés d'argument sont utilisées pour la génération de la chaîne d'arguments requise dans l'appel SQLJ ou db2sqljcustomize. Exemple :
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}

```

Définition d'archive SCLM (exemple ASTSQLJ) :

```

*
*
LKED J2EEOBJ          * J2EE Build translator
*
* Source to include in build
*
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
*
* Build script and generated outputs
*
SINC ASTSQLJ J2EEBLD * J2EE JAR Build script
OUT1 *      J2EEJAR
LIST *      J2EELIST

```

Script de génération AST SCLM (exemple 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"/>

<!-- spécifiez l'emplacement d'exécution du bin JAVA -->
<property name="JAVA_BIN" value="/u/java/J5.0/bin"/>

<!-- spécifiez l'emplacement du fichier db2jcc.jar à inclure dans le chemin
d'accès aux classes db2sqljcustomize -->
<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"/>

<!-- spécifiez les arguments des propriétés globales ci-après pour

db2sqljcustomize -->
<property name="db2sqljcustomize.arg" value="-automaticbind NO -onlinecheck YES
-bindoptions "ISOLATION(CS)" -genDBRM"/>

<!-- spécifiez l'emplacement de la routine d'exécution db2sqljcustomize BWBSQLC -->
<property name="db2sqljcustomize.exec" value="/u/SCLMDT/bwbsqlc.rex&cdqg;/>

<!-- Vous trouverez ci-après le nom du programme utilisateur à exécuter dans le cadre
de la procédure db2sqljcustomize. Il met à jour de manière dynamique un fichier
de propriétés à utiliser comme entrée de la commande db2sqljcustomize -->
<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">
  <!-- Le code ci-après envoie les propriétés dans un fichier de propriétés

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>

```

Source Java dans les fichier d'archive

Par défaut, la source Java est copiée dans l'espace de travail USS au format ASCII avant que les générations de compilation ne soient effectuées. Cela se produit même si la source Java est stockée dans SCLM au format EBCDIC.

Si l'utilisateur demande que la source Java soit incluse dans le fichier d'archive (JAR), la source est par défaut au format ASCII. Il est toutefois possible de configurer les scripts de génération de sorte que la source Java soit copiée dans l'espace de travail et compilée au format EBCDIC pour que, si une inclusion de la source Java est souhaitée, cette dernière soit au format EBCDIC.

Pour inclure la source Java au format EBCDIC, les scripts de génération suivants doivent être modifiés en conséquence, comme suit :

- Dans le principal script XML de génération, ajoutez la ligne `<property name="ASTJAVA_ENCODING" value="EBCDIC"/ >` et supprimez le mot clé `excludes` de la routine de mise à jour du fichier JAR, comme suit : `jar update="true" destfile="{JAR_FILE_NAME}"> <fileset dir="." excludes="**/*.java"/>`.
- Dans le script d'exécution AST référencé par le mot clé `AST_SHSCRIPT`, supprimez le mot-clé `-Dfile.encoding=ISO8859-1 A_parm =`
'-Dfile.encoding=ISO8859-1 -Xnoargsconversion'

SCENARIO D'UTILISATION : 'PlantsByWebSphere'

WebSphere Application Server contient un certain nombre d'exemples de projet. Ces derniers peuvent être déployés à l'aide de fichiers EAR préconfigurés ou assemblés à l'aide d'une procédure de génération ANT. Vous trouverez ci-après un exemple d'utilisation de SCLM Developer Toolkit pour restituer ce projet dans SCLM et le générer/déployer sous z/OS à l'aide de la procédure de génération AST. PlantsByWebSphere est un projet J2EE qui implémente un magasin de vente de plantes en ligne. Il est composé des quatre composants suivants :

- Enterprise Archive (PBWProject)
- Enterprise JavaBean (PlantsByWebSphereEJB)
- Web Archive 1 (PlantsByWebSphereWEB)
- Web Archive 2 (PlantsGalleryWEB)

Ce document décrit un scénario dans lequel cette source est placée dans une structure de projets Eclipse, restituée dans Developer Toolkit, puis générée et déployée sur l'hôte.

1. TACHE D'ADMINISTRATION : Ajoutez les types suivants à la définition de projet :
 - PLANTEAR - Type du fichier EAR
 - PLANTEJB - Type du fichier EJB
 - PLANTWE1 - Type du premier fichier WAR
 - PLANTWE2 - Type du second fichier WAR
2. TACHE D'ADMINISTRATION : Régénération du projet (travail de soumission).
3. TACHE D'ADMINISTRATION : Attribution des fichiers suivants :
 - <HLQ>.<DEVGROUPE>.PLANTEJB
 - <HLQ>.<DEVGROUPE>.PLANTEAR (large)
 - <HLQ>.<DEVGROUPE>.PLANTWE1 (large)
 - <HLQ>.<DEVGROUPE>.PLANTWE2 (large)
4. Recherchez le code source de PlantsByWebSphere dans l'installation de WAS. (~root/AppServer/samples/src/PlantsByWebSphere)
5. Démarrez le produit Eclipse compatible Developer Toolkit de votre choix.
6. Importez les 4 composants dans 4 projets de votre espace de travail Eclipse.
7. Vérifiez que les fichiers .project et .settings sont inclus et définis correctement pour chaque projet.
8. Définissez les paramètres du compilateur et les autres fichiers de configuration spécifiques à Websphere. (Projet... Propriétés)
 - Spécifiez 1.4 comme facette Java.
 - Vérifiez que le raccord d'exécution WAS approprié est associé au projet

9. Ajouter à SCLM (Equipe->Ajouter à SCLM sur le noeud Projet)

a. Les langages à utiliser sont les suivants :

- Images – J2EEBIN
- Fichiers XML – J2EEPART
- Paramètres – J2EEPART
- Code source Java – JAVA

b. Les types à utiliser sont les suivants :

- Fichiers EAR – PLANTEAR
- Fichiers EJB – PLANTEJB
- Fichiers WAR 1 – PLANTWE1
- Fichiers WAR 2 – PLANTWE2

c. Définition d'archive (dernière page de l'assistant Ajouter à SCLM)

- Chaque composant reçoit une définition d'archive avec des mots clés J2EE et un script de génération AST.
- Chaque définition d'archive EAR doit inclure les définitions d'archive des autres composants.

DEFINITION D'ARCHIVE PLANTEAR (exemple) :

```
*
*
LKED J2EE0BJ          * J2EE Build translator
*
* Source to include in build
*
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
*
* Build script and generated outputs
*
SINC PLANTEAR J2EEBLD  * J2EE EAR Build script
OUT1  *          J2EEEAR
LIST  *          J2EELIST
```

d. Scripts de génération, avec les modèles de sortie suivant à nommer :

- PlantsByWebSphere.ear (PLANTEAR)
- PlantsByWebSphereEJB.jar (PLANTEJB)
- PlantsByWebSphere.war (PLANTWE1)
- PlantsGallery.war (PLANTWE2)

SCRIPT DE GENERATION PLANTEAR (exemple) :

```
<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>
```

Pour PLANTWE1, le fichier EJB doit se trouver sur le chemin d'accès aux classes lors de la phase de génération. Par conséquent, modifiez le script de génération de PLANTWE1, pour ajouter une propriété CLASSPATH_JARS_FILES, avec la valeur "PlantsByWebSphereEJB.jar".

10. Générez le projet (via la définition d'archive PLANTEAR)

11. Exécutez le déploiement :

- a. Préparez le script de l'action de déploiement (deploy_ben.jacl). Ce déploiement nécessite de créer une source de données/un connecteur/un fournisseur de messagerie dans WAS et requiert donc un script de déploiement personnalisé.

```
#-----
#
# Exemple de script JACL pour le déploiement d'application J2EE
#
#-----
#
# IBM SCLM Developer Toolkit utilise l'outil wsadmin d'IBM

WebSphere Application
# Server pour déployer les applications J2EE dans WAS, sous z/OS. L'outil
# wsadmin requiert un script JACL pour assister la procédure de déploiement.
# Par conséquent, le script JACL doit être installé sous UNIX Systems Services
# pour que la procédure de déploiement puisse être appelée.
# Cet exemple de script JACL ne doit nécessiter aucune personnalisation.

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 {[length $app] == 0} {
    puts "deploy: Error -- No application specified."
    return
}

#-----
#   -- was the appname supplied
#-----
if {[length $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 temp1Name    "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 $temp1Name $provName

#-----
# Create Datasource
#-----

set temp1Name    "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 $temp1Name $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. Générez le script de propriétés sur le système frontal et lancez le déploiement. (Déployer le squelette)
12. Vérifiez les messages de déploiement. En cas de succès, consultez le projet sur votre serveur WAS.

Annexe E. BUILD FORGE et SCLM

Cette section couvre les questions de mise en oeuvre et de configuration qui permettent d'accéder à Build Forge et de l'utiliser. Elle inclut des exemples de personnalisation et de test pour les générations et les promotions SCLM.

Présentation

Le serveur de console Build Forge se trouve généralement sur une plateforme Windows et doit être configuré avec un appel de service SCLM dans la zone de projet de la console. Au démarrage, ce projet configuré SCLM se connecte au serveur de l'agent Build Forge sous z/OS qui doit déjà se trouver en cours d'exécution. Le plug-in Build Forge permet de démarrer les projets de console à partir de l'environnement RDz en communiquant le serveur de console via une connexion socket. La sortie des exécutions de projet terminées est également accessible à partir du plug-in RDz Build Forge au sein de RDz.

Conditions préalables

- Plug-in Build Forge V7.1 installé dans RDz 7.6
- Console/serveur Build Forge version V7.1
- Agent Build Forge pour z V7.1 (src-bfagent-7.1.1.0-0022.tar.gz ou version ultérieure)
<http://www-01.ibm.com/support/docview.wss?rs=3099&uid=swg24016541>

Remarque : Il est important d'utiliser la version correcte du plug-in Build Forge avec la version correcte du serveur et de l'agent z/OS. Les versions compatibles du plug-in et de l'agent z sont livrées dans le package distribué du serveur Build Forge.

Pour télécharger la version correcte du plug-in, effectuez une mise à jour à partir du site du serveur http://<console_host_name>/prism/eclipse/updateSite/site.xml

Comment appeler l'agent Build Forge sous z/OS

Démarrez l'agent Buildforge sous z/OS. Le port par défaut de l'agent est 5555. Par exemple :

```
bfagent -s -f /répertoire_installation/bfagent-7.1.1.007/src/bfagent.conf
lancez la console BF -> accédez aux serveurs
nom d'hôte
(nomhôte:port)
tester la connexion (à l'aide d'une authentification par ID z/OS)
```

Exemple de réponse :

```
Test de l'agent démarré
Hôte:nomhôte Port:5555
Version de l'agent : 7.1.1.007
Authentification : userid
Plateforme : os/390 18.00 03
Test opérationnel : OK
```

Test de l'agent terminé (durée 9s)

Configurez un projet dans la console Build Forge et lancez l'exécution.

Configuration du serveur de console Build Forge

La configuration minimale suivante est nécessaire pour activer la fonction SCLM au sein de Build Forge.

1. Serveur (Configurez le serveur afin qu'il pointe vers l'agent Build Forge for System z)
 - a. Configurez les autorisations du serveur avec un ID utilisateur/mot de passe z/OS. Cliquez sur **Serveur -> Auth. serveur**

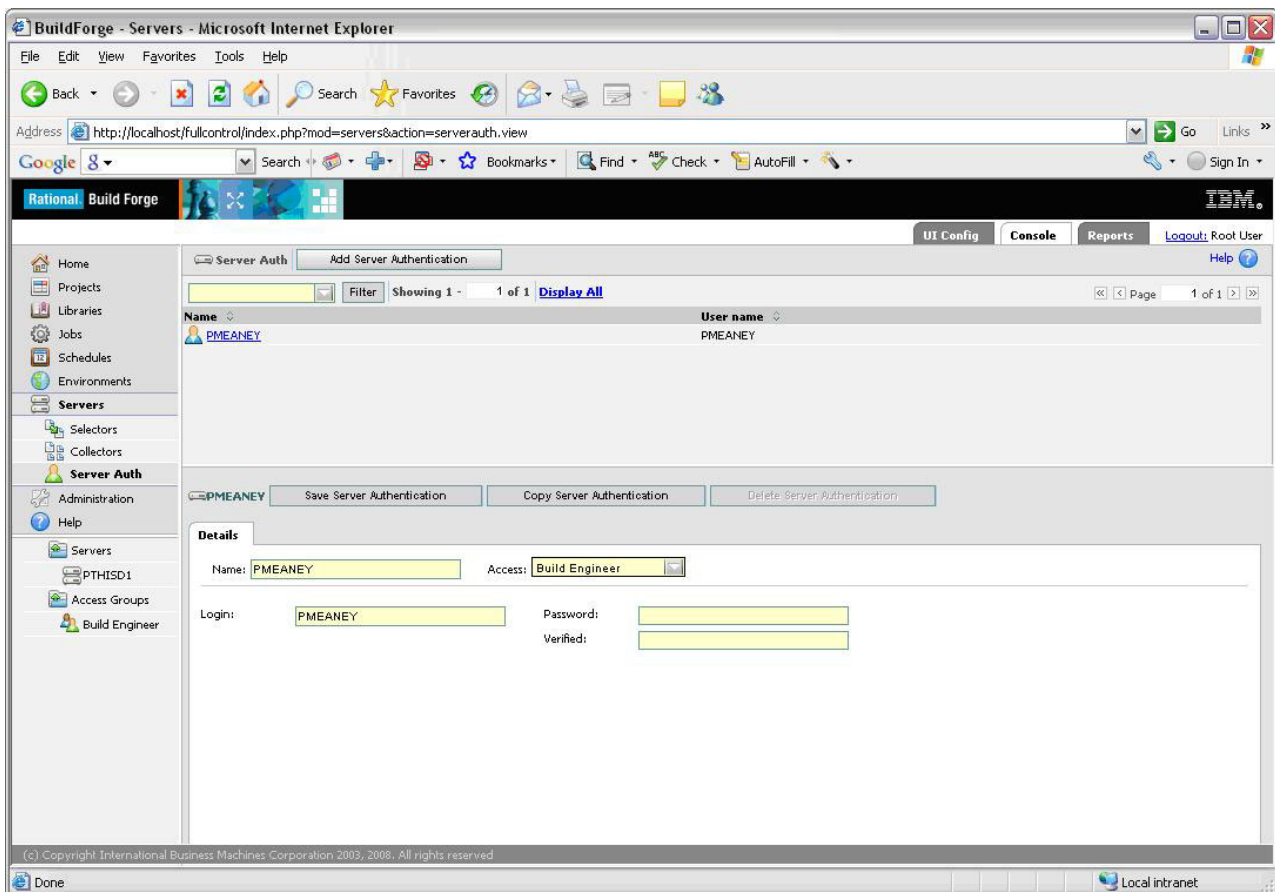


Figure 27. Autorisation du serveur

L'ID utilisateur doit être un ID utilisateur et un mot de passe valides sur le système z/OS où l'agent s'exécute et où vous souhaitez exécuter le service SCLM. Le groupe d'accès est un groupe d'accès existant ou le groupe d'accès par défaut résidant sur le serveur de console avec les droits appropriés : **Administration -> Groupe d'accès**.

- b. Définissez la configuration principale du serveur z/OS. (Cliquez sur **Serveurs**.)

NAME : Nom unique identifiable pour la définition de serveur
PATH : Répertoire du chemin sur l'hôte z/OS où les répertoires

de génération
doivent être créés. Les répertoires de génération sont généralement définis

sous la forme

PATH/nomprojet/BUILD_#/*

Le fichier d'entrée SCLM est créé dans ces répertoires et la réponse

obtenue par l'appel

de service est stockée dans les répertoires en utilisant la convention

de dénomination

établie dans la définition du projet de la console.

HOST : Entrez le nom et le port de l'hôte de destination

(DNS ou adresse IP)

(valeur par défaut de l'agent z/OS : 5555) sous la forme nomhôte:port

ACCESS : Entrez le type d'accès d'autorisation

(Par exemple : Ingénieur)

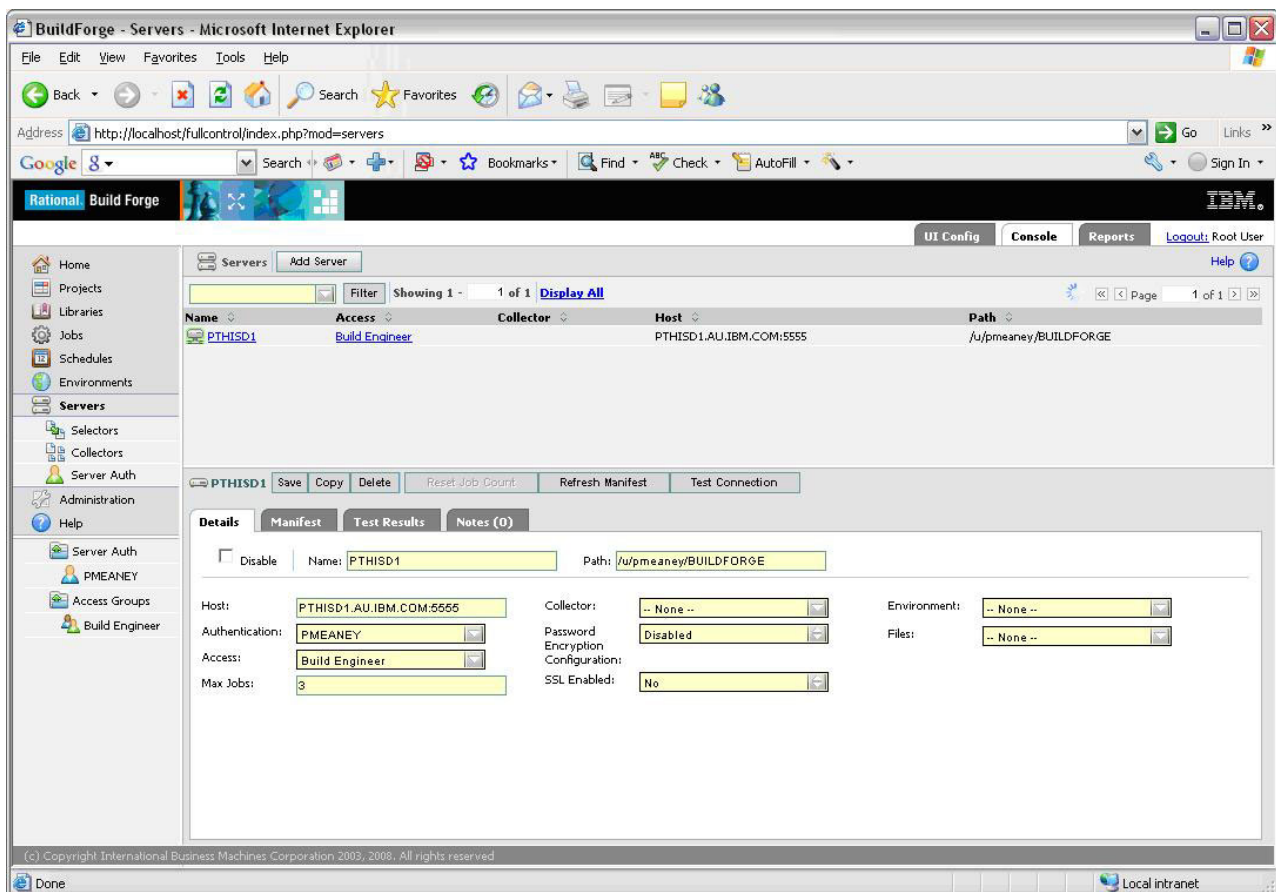


Figure 28. Définition de serveur

Testez la connexion établie avec l'hôte en cliquant sur **Tester la connexion** . Cette opération effectue un test de la connexion établie avec l'agent du serveur exécuté sous z/OS. Le test est réussi s'il apparaît sous la forme suivante :

Test de l'agent démarré
Hôte: pthisd1.au.ibm.com Port: 5555
Version de l'agent : 7.1.1.007
Authentification : IBMUSER
Plateforme : os/390 18.00 03
Test opérationnel : OK
Test de l'agent terminé (durée 9s)

- c. Configurez le principal sélecteur du serveur. Cliquez sur **Serveurs -> Sélecteur**.
Affectez BF_NAME au nom du serveur. (Par exemple : PTHISD1)

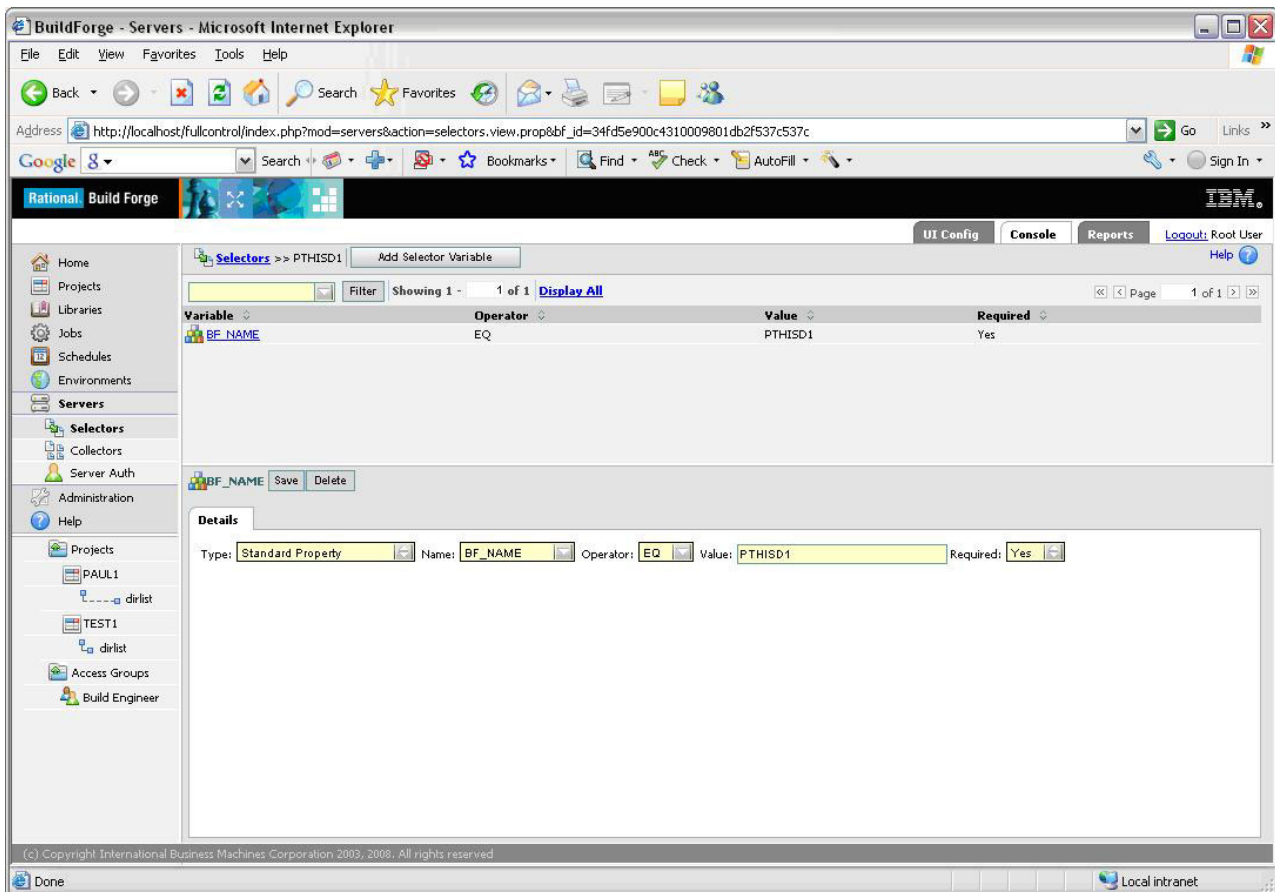


Figure 29. Définition du sélecteur de serveur

2. Configuration des variables d'environnement (Configurez les variables d'environnement des outils de développement SCLM pour le projet SCLM.)

- a. Cliquez sur **Environnements** – Créez un nom pour la liste de variables d'environnement DT SCLM DT. Par exemple : SCLMDT

Configurez les variables d'environnement SCLM DT suivantes :

STEPLIB : Fichier STEPLIB où les modules SCLM Developer Toolkit résident. Il se trouve dans le fichier RDz SFEKLOAD.

(Par exemple : RD4Z.V760.SFEKLOAD) Ce type d'action dans la définition doit être APPEND.

_CMDSERV_BASE_HOME : Répertoire de base où la passerelle ISPF est installée (par exemple : /usr/lpp/ispf).

_CMDSERV_CONF_HOME : Répertoire de configuration de la

passerelle ISPF

(par exemple : /etc/ispf).

_CMDSERV_WORK_HOME : Répertoire de travail de la passerelle

ISPF

(par exemple : /var/ispf).

_SCLMDT_CONF_HOME : Répertoire de configuration de SCLM DT au sein de RDz. (Par exemple : /etc/rd4z760/sclmdt).

_SCLMDT_WORK_HOME: Répertoire de travail de SCLM DT dans RDz. En général, il s'agit du même répertoire que celui

de la variable

_CMDSERV_WORK_HOME

(Par exemple : \$_CMDSERV_WORK_HOME).

PATH : Chemins de répertoire suivants : RDz bin, le répertoire bin de _CMDSERV_BASE_HOME et le répertoire de script sclmdt de RDz.

Doit correspondre au type d'action APPEND.

Par exemple :

/apc/trdz750/usr/lpp/rdz/bin:/etc/rd4z760/sclmdt/CONFIG/script:
\$_CMDSERV_BASE_HOME/bin

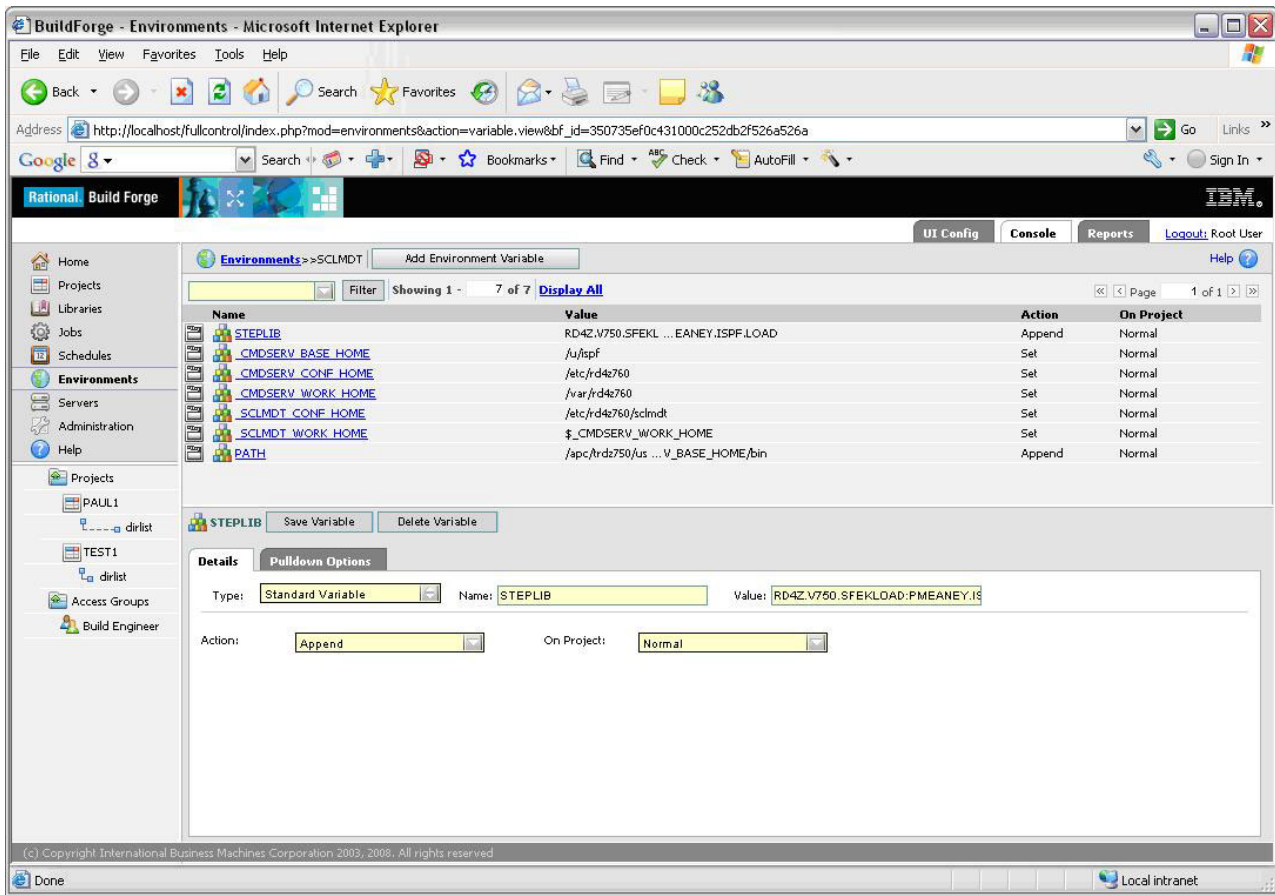


Figure 30. Définitions des variables d'environnement

3. Configuration de projets SCLM dans Build Forge

Vous devez à présent configurer des projets SCLM pour les différentes demandes de service SCLM. Les exemples ci-après s'appliquent aux services de génération et de promotion SCLM qui sont généralement les principaux services adaptés à la planification de travaux Build Forge. Les exemples de script de projet reposent sur le format d'API XML SCLM indiqué dans le document *SCLM Developer Toolkit Administrators Guide*.

- Configurez d'abord un projet SCLM en cliquant sur **Projets -> Ajouter projet**.

NAME : Entrez le nom de ce projet.

(Par exemple : SCLM build sample1)

SELECTOR : Entrez le nom de sélecteur configuré dans la table de sélecteurs Server (comme à la section précédente 1c, à la page 126).

ENVIRONMENT : Entrez le nom de l'environnement configuré pour ce projet SCLM qui contient les variables d'environnement SCLM (comme indiqué à la section précédente 2a, à la page 126).

Par exemple : SCLMDT

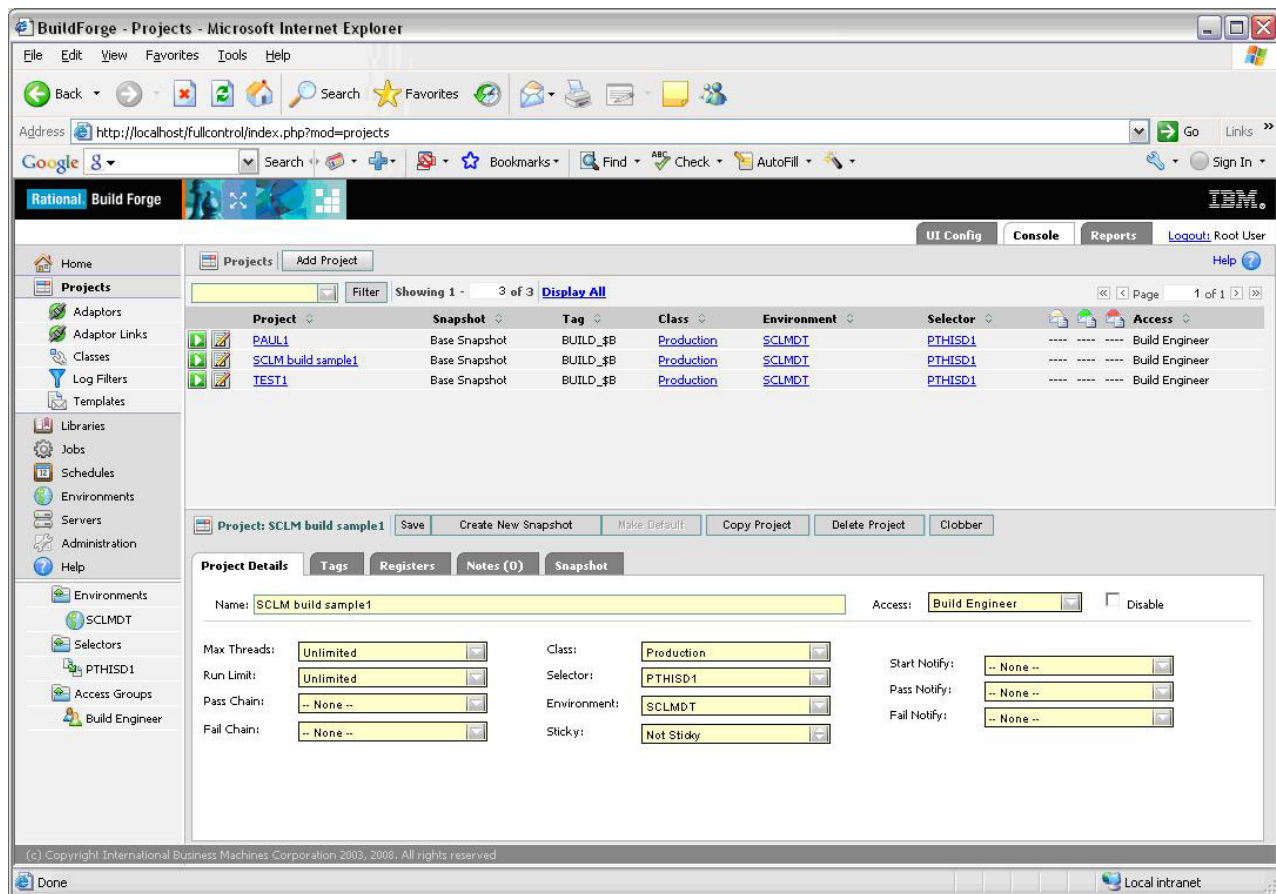


Figure 31. Exemple de configuration d'un projet pour SCLM (SCLM build sample1)

b. Ajoutez à présent une étape de génération à ce projet SCLM.

L'exemple suivant est un exemple de génération configuré distribué dans le fichier SAMPLIB de l'hôte (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

```

Figure 32. ** SCLM Build sample1 **

Configurez l'exemple ci-dessus avec les options de projet, de groupe, de type, de membre et de génération SCLM, comme indiqué dans l'ANNEXE C de la fonction de génération, à la section relative à l'API DT SCLM.

Ajoutez cet exemple configuré à la première étape de l'exemple Project SCLM Build sample1.

Projets -> SCLM Build sample1 -> ajouter l'étape
NAME : Nom que vous avez choisi pour l'étape
COMMAND : Incluez l'exemple configuré ci-dessus (BWBBFBLD) dans cette zone

Toutes les autres zones doivent conserver les valeurs par défaut. (Cette étape hérite des paramètres de l'environnement du projet principal.)

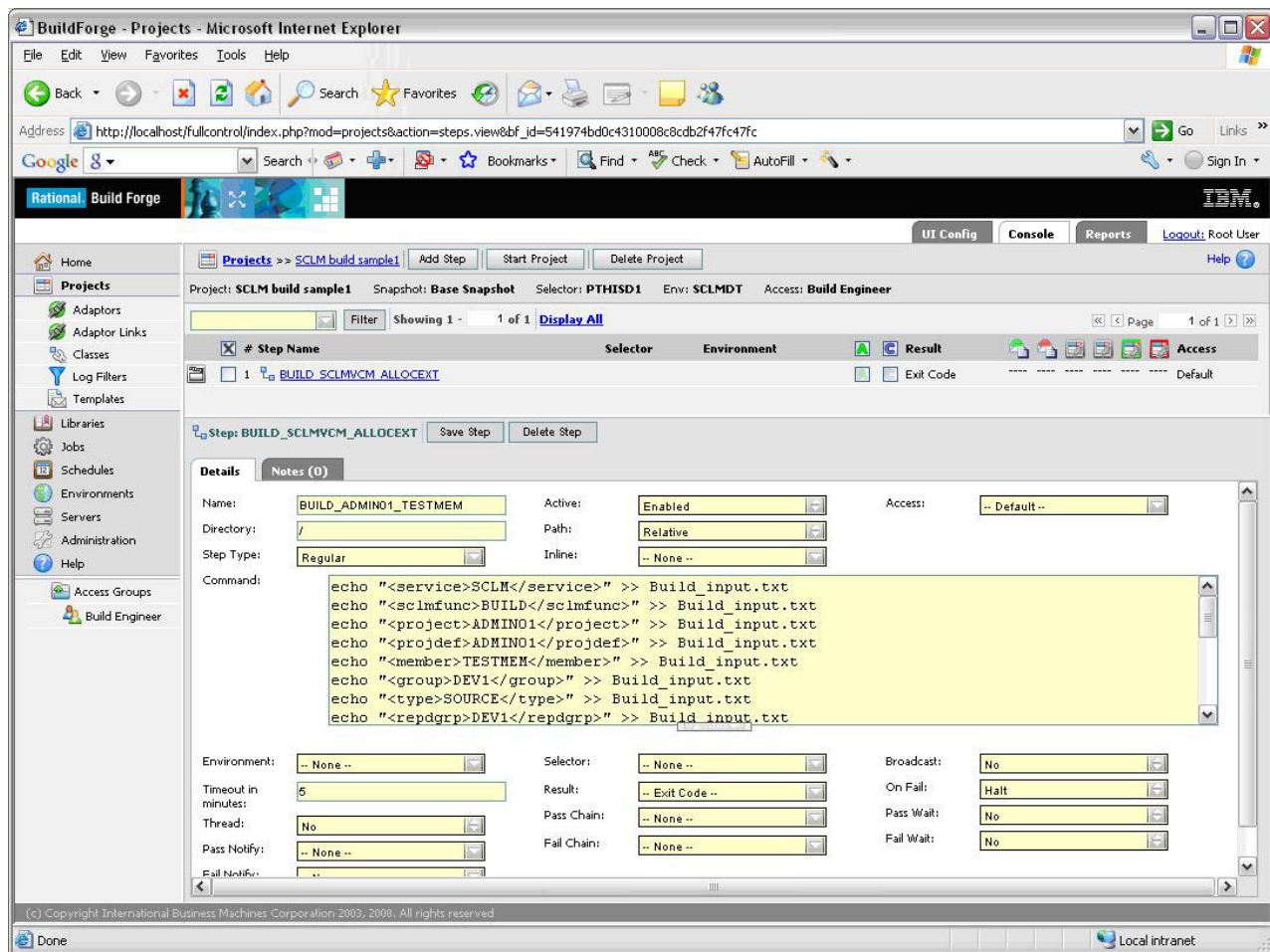


Figure 33. Etape de l'exemple de projet dans SCLM build sample1

- c. Exécutez l'exemple "SCLM Build sample1" du projet de génération SCLM. Cliquez sur **Projets -> SCLM Build sample1 -> Démarrer projet**. La sortie de la génération est renvoyée dans la console **TRAVAUX -> Balise de génération**.

La sortie de la demande et de la réponse sont également stockés dans le répertoire z/OS Unix Systems Services défini par la variable PATH indiquée dans la configuration du serveur. Dans l'exemple précédent, pour le serveur PHISD1 with PATH=/u/IBMUSER/BUILDFORGE, les fichiers de demande et de réponses sont stockés dans le répertoire:

```
/u/IBMUSER/BUILDFORGE/SCLM_Build1_sample/BUILD_1
: >ls
Build_input.txt  Build_output.txt
```

Les fichiers de demandes et de réponses ci-dessus peuvent être renommés lors de l'étape de génération personnalisée.

Exemple de promotion SCLM

Les étapes de la section précédente peuvent être répétées pour créer un projet de promotion dans Build Forge. Remplacez le script de génération par un exemple de script de promotion dans la zone "COMMAND" du projet. L'exemple suivant est un exemple de promotion configuré dans le fichier SAMPLIB de l'hôte (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
```

Figure 34. **** Exemple "SCLM Promote sample1" ****

Configurez l'exemple précédent avec vos options de projet, de type, de membre et de promotion SCLM, comme indiqué dans la fonction Annexe C, «API de SCLM Developer Toolkit», à la page 65 for Promote.

Remarques sur la documentation pour IBM Rational Developer for System z

© Copyright IBM Corporation - 2009

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.*

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues auprès du IBM Intellectual Property Department de votre pays ou par écrit à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni à aucun pays dans lequel il serait contraire aux lois locales : LE PRESENT DOCUMENT EST LIVRE "EN L'ETAT" SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFAÇON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

Intellectual Property Dept. for Rational
Software
IBM Europe Middle-east Africa
3039 Cornwallis Road, PO Box 12195
Research Triangle Park, NC 27709
U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans cette documentation et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'IBM Customer Agreement, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

Licence de copyright

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les exemples de programme sont fournis "EN L'ETAT", sans garantie d'aucune sorte. IBM ne sera en aucun cas responsable des dommages liés à l'utilisation de ces exemples de programme.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp. dans la plupart des juridictions du monde. Les autres noms de produit et service sont des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web "Copyright and trademark information" à www.ibm.com/legal/copytrade.shtml.

Rational est une marque d'International Business Machines Corporation et de Rational Software Corporation aux Etats-Unis et/ou dans certains autres pays.

Intel et Pentium sont des marques d'Intel Corporation aux Etats-Unis et/ou dans certains autres pays.

Microsoft, Windows et le logo Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

Java et toutes les marques et logos incluant Java sont des marques de Sun Microsystems, Inc. aux Etats-Unis et/ou dans certains autres pays.

UNIX est une marque enregistrée de The Open Group aux Etats-Unis et/ou dans certains autres pays.

Index

Caractères spéciaux

[profil sérialisé], liaison 45
\$GLOBAL 24

A

API, SCLM Developer Toolkit 65
API de SCLM Developer Toolkit 65
ARCHDEF 8, 58
attributs de fichier, recommandés 7
attributs pour types de fichier standard, recommandés 7
authentification, clé publique 22
authentification de clé, publique 22
authentification de clé publique 22
AUTHUPD - Modification du code des droits d'accès 69

B

BROWSE - Exploration d'un membre 70
BUILD - Création d'un membre 70
BUILD FORGE et SCLM 123
BWBC9DTJ 17
BWBCRON1 51
BWBDEPJ1 18
BWBDEPJ2 18
BWBDEPJ3 18
BWBEARA 17
BWBEJBA 17
BWBJAVAA 17
BWBSQLB 17
BWBSQLBE 17
BWBTANX 18
BWBWEBA 17

C

CLASSPATH_JARS 14
CLASSPATH_JARS_FILES 14
code des droits d'accès, AUTHUPD 69
combinaisons de remplacements de fichier BIDIPROP, exemple d'utilisation 33
combinaisons de remplacements de fichier TRANSLATE.conf, exemple d'utilisation 32
commandes, schéma XML pour SCLMDT 66
commandes SCLMDT, schéma XML 66
Comparaison de JDBC et SQLJ 36
concepts, JAVA/J2EE 59
concepts, SCLM 57
concepts JAVA/J2EE 59
copie d'un fichier JAR, JARCOPY 77

D

DB2 35
db2sqljcustomize.* properties 41
DBRMLIB 9
définitions, langage SCLM 5
définitions de langage 5
DELETE - Suppression d'un membre 72
dénomination, fichiers 57
dénomination des fichiers 57
dépendances, CLASSPATH 14
Dépendances CLASSPATH 14
déploiement, de SCLM sur UNIX System Services 22
déploiement, SCLM Developer Toolkit 20
déploiement, sécurisé 22
déploiement, WebSphere Application Server 21
déploiement de WebSphere Application Server 21
déploiement sécurisé 22
déploiement sur UNIX System Services, de SCLM 22
DEPLOY - Déploiement d'un fichier EAR J2EE 72

E

EDIT - Edition d'un membre 73
exécution, travaux de génération CRON 52
exécution des travaux, génération CRON 52
exécution des travaux de génération, CRON 52
exécution des travaux de génération CRON 52
Exemples 88
exemples, ARCHDEF J2EE 11
exemples, sclmdt_request.xml 88
exemples, sclmdt_response.xml 90
exemples, travaux de génération CRON 53
exemples, xmlbld.java 88
exemples d'ARCHDEF, J2EE 11
exemples d'ARCHDEF J2EE 11
exemples de script J2EE 15
exemples de travaux, génération CRON 53
exemples de travaux de génération, CRON 53
exemples de travaux de génération CRON 53
extraction de LRECL du fichier, LRECL 78
extraction des groupes d'un projet, PROJGRPS 80
extraction des informations sur un projet, PROJINFO 80

extraction du statut d'un travail par lots, JOBSTAT 78

F

fichier, extraction de LRECL 78
fichier EAR, DEPLOY 72
fichier J2EE EAR, DEPLOY 72
fichiers pour JAVA/J2EE 7
fonctions et paramètres, demandes 68
FORGE et SCLM, BUILD 123
format, appel 65
format, fonction 68
format d'appel 65
format des fonctions 68
formats, membres SCLM 9
formats des membres, SCLM 9
formats des membres SCLM 9

G

générations et promotions, lancées via le service CRON 51
générations et promotions lancées via le service CRON 51
groupes, extraction 80

H

historique, versions 86
historique des versions, VERHIST 86

I

importation d'un projet à partir de SCLM, J2EEIMP 74
INCL 10
INCLD 10
INFO - Informations sur le statut du membre 74
informations d'un membre, UPDATE 84
informations sur le statut, INFO 74
informations sur le statut du membre, INFO 74
informations sur un projet, extraction 80
installation, produit 1
installation du produit 1

J

J2EEANT 5
J2EEBIN 6
J2EEBLD 8
J2EEEAR 9
J2EEIMP - Importation d'un projet à partir de SCLM 74
J2EEJAR 9
J2EELIST 8

J2EEMIG - Migration d'un projet dans SCLM 75
 J2EEMIGB - Migration par lots d'un projet dans SCLM 77
 J2EEOBJ 6
 J2EEPART 6
 J2EEWAR 9
 JARCOPY - Copie d'un fichier JAR 77
 JAVA 6
 JAVA/J2EE, fichiers SCLM 7
 JAVABIN 7
 JAVACLAS 9
 JAVALLIST 8
 JDBC 35
 JDBC et SQLJ, comparaison 36
 JOBSTAT - Extraction du statut d'un travail par lots 78

L

Langage 58
 liaison 39
 liaison [Module d'interrogation de base de données] 43
 liaison [profil sérialisé] 45
 LIST 10
 liste, fonctions 68
 liste des fonctions 68
 liste des membres et des fichiers non SCLM, MIGDSN 79
 liste des membres et des fichiers NON-SCLM, MIGPDS 79
 LKED 10
 LRECL - Extraction de LRECL du fichier 78

M

mappage de projets J2EE à SCLM 18
 mappage de projets J2EE à SCLM, recommandations 19
 membre, BROWSE 70
 membre, BUILD 70
 membre, DELETE 72
 membre, EDIT 73
 membre, PROMOTE 80
 membre, SAVE 83
 membre, UNLOCK 84
 membres, liste des membres et des fichiers non SCLM 79
 membres et fichiers, liste des membres et des fichiers non SCLM 79
 Membres et fichiers non SCLM, liste 79
 MIGDSN - Liste des membres et des fichiers non SCLM 79
 MIGPDS - Liste des membres et des fichiers NON-SCLM 79
 migration d'un projet, J2EEMIG 75
 migration d'un projet (par lots), J2EEMIGB 77
 migration par lots d'un projet, J2EEMIGB 77
 module d'interrogation de base de données, liaison 43
 module d'interrogation de base de données, présentation 37

O

objets de génération, JAVA/J2EE 4
 objets de génération JAVA/J2EE 4
 options, autres options de déploiement 23
 options, spécifiques au projet et SITE 28
 options de déploiement, autres 23
 options de stockage, ASCII ou EBCDIC 23
 options de stockage ASCII ou EBCDIC 23
 options de stockage EBCDIC 23
 options spécifiques au projet 28
 Options spécifiques au projet et SITE 28
 OUT1 10

P

paramètres, fonctions des demandes 68
 paramètres et fonctions des demandes 68
 Personnalisation 39
 personnalisation de SCLM Developer Toolkit 3
 préparation, programme SQLJ 38
 préparation du programme, SQLJ 38
 préparation du programme SQLJ 38
 présentation, SCLM 57
 procédure de génération 40
 profil sérialisé 37
 programme de traduction, récapitulatif technique de SCLM 61
 projets J2EE, mappage à SCLM 18
 projets J2EE, recommandations pour le mappage 19
 PROJGRPS - Extraction des groupes d'un projet 80
 PROJINFO - Extraction des informations sur un projet 80
 PROMOTE - Promote member 80
 promotions, générations lancées via le service CRON et 51
 propriétés, db2sqljcustomize.* 41
 propriétés, SCLM 58
 propriétés, sqlj.* 41

R

rapport, DBUTIL 82
 rapport DBUTIL, REPUTIL 82
 rapport de projet, création 82
 récapitulatif de la génération, JAVA/J2EE 3
 récapitulatif de la génération JAVA/J2EE 3
 récapitulatif du programme de traduction SCLM, technique 61
 Récapitulatif technique du programme de traduction SCLM 61
 remarque sur la génération, JAVA/J2EE 24
 remarque sur la génération JAVA/J2EE 24
 remplacements, exemple d'utilisation de combinaisons de BIDIPROP 33

remplacements, exemple d'utilisation de combinaisons de TRANSLATE.conf 32
 remplacements BIDIPROP, exemple d'utilisation de combinaisons 33
 remplacements TRANSLATE.conf, exemple d'utilisation de combinaisons 32
 REPORT - Création d'un rapport de projet 82
 REPUTIL - Rapport DBUTIL 82

S

SAVE - Sauvegarde d'un membre 83
 schéma des commandes SCLMDT, XML 66
 Schéma XML des commandes SCLMDT 66
 SCLM, BUILD FORGE 123
 sclmdt_request.xml 88
 sclmdt_response.xml 90
 script, utilisateur personnalisé 42
 script de génération, personnalisation 41
 script utilisateur, personnalisé 42
 script utilisateur personnalisé 42
 scripts, exemple J2EE 15
 sécurité, SCLM 47
 SINC 10
 spécifications, de STEPLIB et PATH 52
 spécifications de PATH, et de STEPLIB 52
 spécifications de STEPLIB et PATH 52
 SQL 35
 SQLJ 7
 sqlj.* properties 41
 SQLJ, qu'est-ce que SQLJ 36
 SQLJ et JDBC, comparaison 36
 SQLJSER 9
 squelettes de génération, XML Ant JAVA/J2EE 16
 squelettes de génération XML, Ant JAVA/J2EE 16
 squelettes de génération XML Ant, JAVA/J2EE 16
 squelettes de génération XML Ant JAVA/J2EE 16
 statut, extraction du statut d'un travail par lots 78
 statut d'un travail, extraction 78
 statut d'un travail par lots, extraction 78
 structure, projet SCLM 58
 structure d'un projet, SCLM 58
 support, SQLJ 35
 support JAVA/J2EE, traducteurs de langage 5
 support SQLJ 35
 suppression de versions, VERDEL 85

T

table, traduction des noms longs/abrévés 61
 table de traduction, noms longs/abrévés 61
 table de traduction des noms, longs/abrévés 61

- table de traduction des noms
 - longs/abrégés 61
- traducteurs, langage ASCII/EBCDIC 23
- traducteurs, types SCLM DT et 40
- traducteurs de langage,
 - ASCII/EBCDIC 23
- traducteurs de langage, support
 - JAVA/J2EE 5
- traducteurs de langage
 - ASCII/EBCDIC 23
- traducteurs de langage EBCDIC 23
- traducteurs et types SCLM DT 40
- Traduction 38
- traitement, FINDLONG 62
- traitement, FINDSHORT 62
- traitement, IMPORT 64
- traitement, MIGRATE 64
- traitement, TRANSLATE 63
- traitement des enregistrements, traitement
 - multiple des enregistrements de nom
 - long/abrégé 63
- traitement des enregistrements, traitement
 - unique des enregistrements de nom
 - long/abrégé 62
- traitement des enregistrements de nom,
 - traitement multiple des enregistrements
 - de nom long/abrégé 63
- traitement des enregistrements de nom,
 - traitement unique des enregistrements
 - de nom long/abrégé 62
- traitement des enregistrements de nom
 - long/abrégé, multiple 63
- traitement des enregistrements de nom
 - long/abrégé, unique 62
- traitement FINDLONG 62
- traitement FINDSHORT 62
- traitement IMPORT 64
- traitement MIGRATE 64
- traitement TRANSLATE 63
- Type 57
- types, Java/J2EE 9
- types, SCLM 7
- types et traducteurs, SCLM DT 40
- types Java/J2EE 9

U

- UNLOCK - Déverrouillage d'un
 - membre 84
- UPDATE - Mise à jour des informations
 - d'un membre 84
- utilisation de combinaisons de
 - remplacements de fichier BIDIPROP,
 - exemple 33
- utilisation de combinaisons de
 - remplacements de fichier
 - TRANSLATE.conf, exemple 32
- Utilitaire de génération, Rational
 - Application Developer for WebSphere
 - Software 97
- utilitaire de génération Rational
 - Application Developer for WebSphere
 - Software 97

V

- variables, définies par le client 13
- variables définies par le client 13
- VERBROW - versions d'exploration 85
- VERDEL - Suppression de versions 85
- VERHIST - Historique des versions
 - SCLM 86
- VERLIST - Versions des listes 86
- VERREC - Versions de restauration 87
- versions, exploration 85
- versions, liste 86
- versions, restauration 87
- versions, suppression 85
- versions d'exploration, VERBROW 85
- versions de restauration, VERREC 87
- versions des listes, VERLIST 86

X

- xmlbld.java 88



Numéro de programme : 5724-T07

SC11-6464-01

