

IKANALM

Integrating IKAN ALM and Mainframes

Cost-effective and easy to implement Enterprise-wide ALM for both mainframe and non-mainframe environments

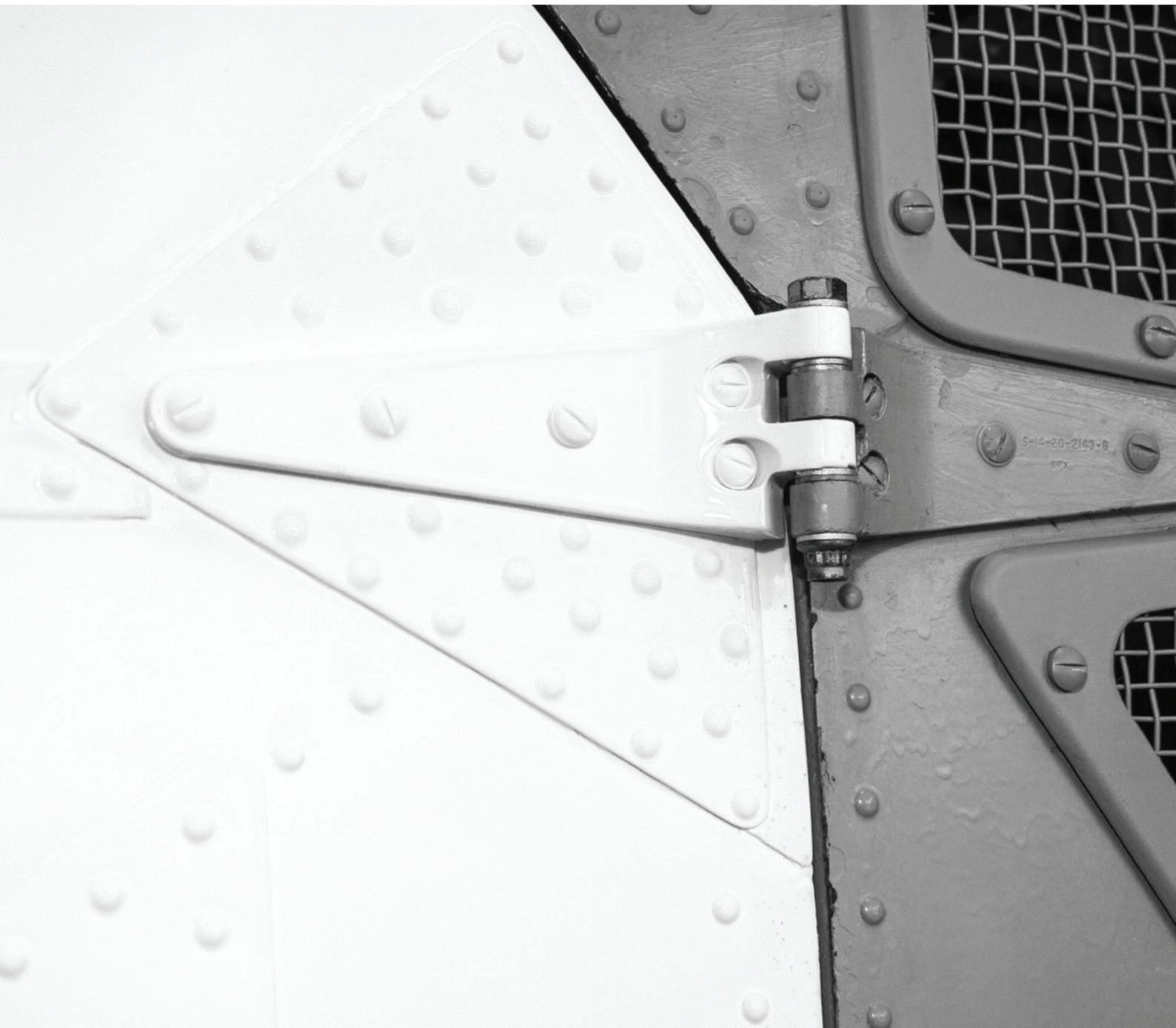


Table of contents

IKAN ALM: The Ideal ALM Solution for the Mainframe	4
IKAN ALM – User’s Point of View.....	4
Step1: Log on and display the Desktop Overview	5
Step 2: Create/update a package and link it to a Project Stream	6
Step 3: Select the required Action	9
Step 4: Create the Level Request.....	9
Alternative Way: Commandline Interface (CLI)	10
Additional information provided by IKAN ALM	11
What happens behind the scene?	14
IKAN ALM –Administrator’s Point of View.....	20
Step1: Create the global Phases.....	20
The Phase concept	20
The IKAN ALM Phase Structure.....	20
The Common Files.....	20
The Resource Files.....	21
The Model files.....	23
The compileCobol_jcl.model	24
An IKAN ALM phase and its usage: the z/OS compile phase	25
An IKAN ALM phase and its usage: the z/OS deployment phase	27
Step 2: Create the IKAN ALM project(s)	28
Step 3: Adapt the lifecycle (if necessary)	29
Step 4: Define the environments and the necessary parameters.....	30
Step 5: Add phases	31
Step 6: Modify the phase parameters	32
Conclusion.....	34
Related Document	34
Appendix I: IKAN ALM Terminology.....	35
Appendix II: CA-ENDEVOR Terminology	36
Appendix III: Serena ChangeMan ZMF terminology	38
Appendix IV: Available z/OS IKAN ALM Phases	39
Appendix V: Migration to IKAN ALM	41
Appendix VI: Sample of z/OS compilation JCL	41

Summary

This technical document is intended for developers, technical people, mainframe or non-mainframe experts, and software architects.

IKAN ALM is a web-based Application Lifecycle Management tool. It combines continuous integration and lifecycle management, offering a single point of control and delivering support for build and deploy processes (manually generated or automated), approval processes, release management, and software lifecycles. IKAN ALM tightly integrates with leading third-party versioning solutions, build and deploy tools, and issue tracking software. It supports both mainframe and non-mainframe systems and, in case of mixed environments, it will handle the dependencies between both systems.

This document aims at explaining how, by using IKAN ALM, you can manage your application lifecycle, be it on mainframe or on distributed systems or on a combination of the two, and how you can easily deploy the developed applications on the mainframe.

We will describe in detail how IKAN ALM works and what the different tasks are for Users and Administrators.

We are confident that after having read this document, you will be convinced of the enormous advantages of putting in place our IKAN ALM solution.

If you would still have questions, do not hesitate to contact our support team at info@ikanalm.com.

Remark: Although IKAN ALM supports many types of mainframes (IBM, Fujitsu, Unisys, Bull,..), we will use IBM z/OS as an example throughout this document.

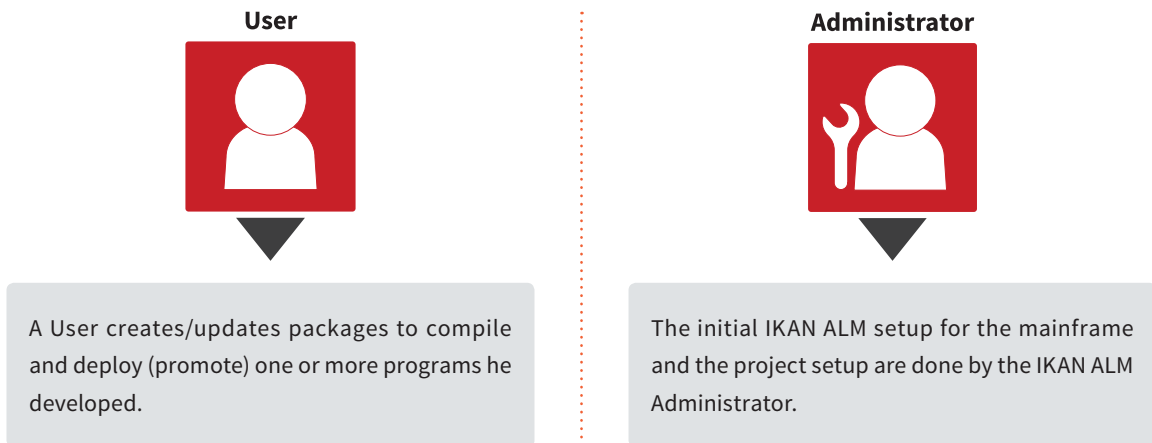


IKAN ALM: The Ideal ALM Solution for the Mainframe

In the following section, we will explain more in detail how IKAN ALM handles the lifecycle to compile and deploy your applications on the mainframe.

Today, traditional mainframe development is already often enhanced with Eclipse-based development to address the requirements of modern end-user applications. The main issue when combining mainframe and distributed development, is how to deploy the developed applications on the mainframe.

We will cover 2 points of view: the IKAN ALM User and the IKAN ALM Administrator.



IKAN ALM – User’s Point of View



Once the initial setup has been done and the projects have been set up by the IKAN ALM Administrator, a User can start using IKAN ALM.

Basically, a User can create a Compile/Build or Deploy action (a “Level Request” in IKAN ALM terminology) in 4 steps:

1. Log on to IKAN ALM and display the Desktop Overview
2. Create or update a package and link it to a Project Stream
3. Select the appropriate Action (Compile/Build or Deploy)
4. Create the Level Request

Once the Level Request is created, a series of information screens will be available to provide additional information on the requested action, allowing following up its status and the results.

Step 1

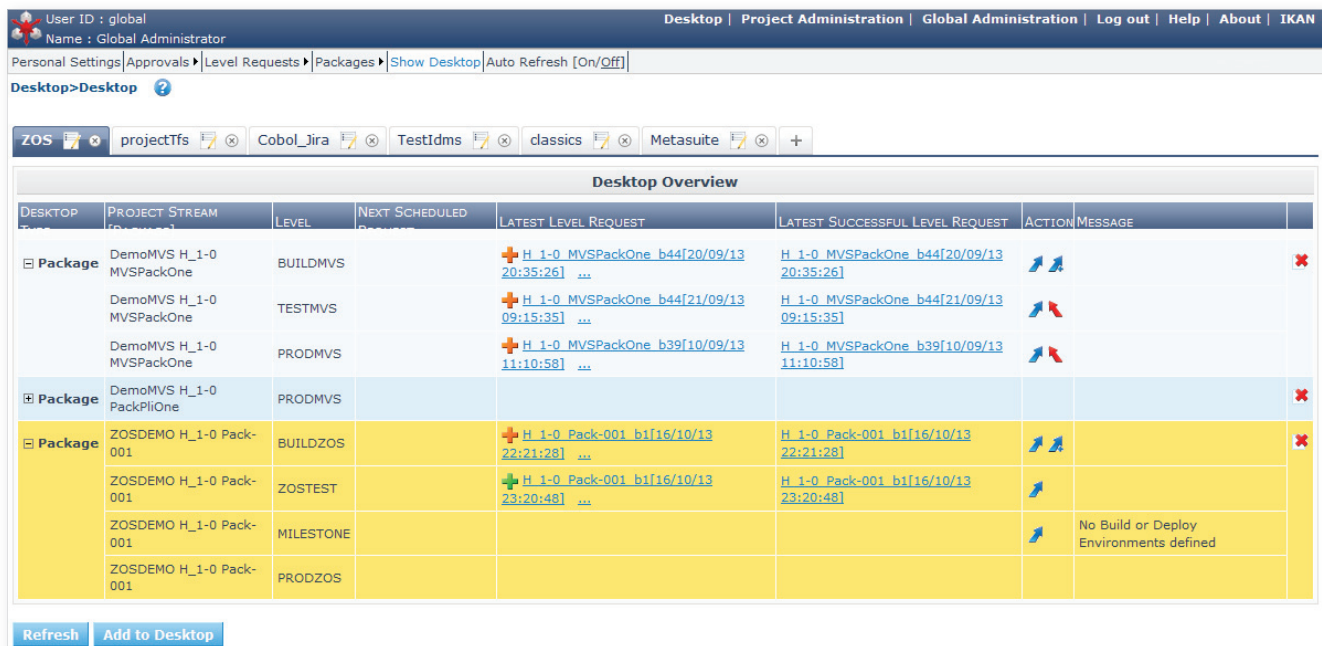
Step1: Log on and display the Desktop Overview



The login screen for IKAN ALM. It features the IKAN ALM logo at the top, with the text "Build B_5-5_0_b37" and "Copyright © 2003-2013 IKAN Development N.V." below it. A "Log in" window contains two input fields: "User ID" and "Password", each with a red asterisk indicating a required field. Below the fields are two buttons: "Log in" and "Reset".

IKAN ALM logon screen

Next, the Desktop Overview will be displayed showing the list of Project Streams or Packages the User is working with. This Desktop can be completely customized.



The screenshot shows the IKAN ALM Desktop Overview interface. The user is logged in as "global" (Global Administrator). The interface includes a navigation bar with "Desktop", "Project Administration", "Global Administration", "Log out", "Help", "About", and "IKAN". Below the navigation bar is a breadcrumb trail: "Personal Settings | Approvals | Level Requests | Packages | Show Desktop | Auto Refresh [On/Off]". The main content area is titled "Desktop Overview" and displays a table of project streams and packages. The table has columns for "DESKTOP", "PROJECT STREAM", "LEVEL", "NEXT SCHEDULED", "LATEST LEVEL REQUEST", "LATEST SUCCESSFUL LEVEL REQUEST", "ACTION", and "MESSAGE".

DESKTOP	PROJECT STREAM	LEVEL	NEXT SCHEDULED	LATEST LEVEL REQUEST	LATEST SUCCESSFUL LEVEL REQUEST	ACTION	MESSAGE
Package	DemoMVS H_1-0 MVSPackOne	BUILDMVS		H_1-0 MVSPackOne_b44[20/09/13 20:35:26]	H_1-0 MVSPackOne_b44[20/09/13 20:35:26]		
	DemoMVS H_1-0 MVSPackOne	TESTMVS		H_1-0 MVSPackOne_b44[21/09/13 09:15:35]	H_1-0 MVSPackOne_b44[21/09/13 09:15:35]		
	DemoMVS H_1-0 MVSPackOne	PRODMVS		H_1-0 MVSPackOne_b39[10/09/13 11:10:58]	H_1-0 MVSPackOne_b39[10/09/13 11:10:58]		
Package	DemoMVS H_1-0 PackPliOne	PRODMVS					
Package	ZOSDEMO H_1-0 Pack-001	BUILDZOS		H_1-0 Pack-001_b1[16/10/13 22:21:28]	H_1-0 Pack-001_b1[16/10/13 22:21:28]		
	ZOSDEMO H_1-0 Pack-001	ZOSTEST		H_1-0 Pack-001_b1[16/10/13 23:20:48]	H_1-0 Pack-001_b1[16/10/13 23:20:48]		
	ZOSDEMO H_1-0 Pack-001	MILESTONE					No Build or Deploy Environments defined
	ZOSDEMO H_1-0 Pack-001	PRODZOS					

At the bottom of the table, there are two buttons: "Refresh" and "Add to Desktop".

Desktop Overview for the z/OS project

The following basic information will be displayed:

- **Project type:** release-based or package-based
- **Project Stream (package) name**
- **Defined Levels:**

A level is a logical step in the application lifecycle. The available levels are: Build (Compile), Test and Production. One or more of each of those levels can be used to define a lifecycle.

- **Next Scheduled Request:**
If a Schedule was assigned to a Level (like in continuous integration) this field contains the execution date and time of the next request.
- **Latest Level Request:** shows the status of the latest request, the VCR tag and timestamp.
- **Latest successful Level Request:** shows the latest successful level request
- **Action:** the available action icons for the Level. When clicking an action button, a level request will be started.
- **Message:** if it is not possible to define a request for a specific level, this message indicates the reason.

The z/OS project we use here as an example is a package-based project for which the following Levels have been defined: a Build Level (BUILDZOS) and some Deploy Levels (ZOSTEST, MILESTONE and PRODZOS). Those Levels are linked to Lifecycle(s), and the Project Stream(s) (i.e., the Head or a Branch) is/are also linked to a Lifecycle.

For mainframe use, a project must be package-based. A package allows the User to select one or several components of a Project Stream which should be built and deployed together, ignoring the other Project Stream components. Such a package has to stay coherent for building and deploying. A Package (of components) will always be linked to a Project Stream (in our example: ZOSDEMO H_1.0 Pack-001) and it will always follow that Project Stream's lifecycle. Also, a package has to progress with the Project Stream's lifecycle independently of possible other packages linked to the same Project Stream.

For distributed release-based projects, on the other hand, all components are built and deployed together.

Step 2

Step 2: Create/update a package and link it to a Project Stream

Before compiling/building, the User has to create a package that contains the sources he wants to compile/build and the copybooks.

1. First of all, the User has to specify the link to the correct Project Stream.

The screenshot shows the 'Create Package' interface. The main dialog has fields for 'Project Stream', 'Name', and 'Description', along with a 'Create' button. An overlaying 'Select Project Stream' dialog is active, featuring a search section with fields for 'Project Name', 'VCR', 'Project Type' (set to 'Package-based'), 'Project Stream Type' (radio buttons for 'Head' and 'Branch'), 'Status', 'Locked' (radio buttons for 'Yes' and 'No'), and 'Show Hidden Project Streams' (radio buttons for 'Yes', 'No', and 'All'). A 'Reset' button is located below the search section. To the right of the search section is a list of project streams: DemoMVS, DEMOZOS, DEVE, DEVE-3, Deve-Tommie, LIVF, MetasuiteWin, MetasuiteZos, ZOSDEMO, and ZOSDEMO 1-0. A 'Select Project Stream' button is at the bottom of this list. A separate rounded rectangle on the right side of the image contains the text 'Select Project Stream'.

2. Next, he has to specify the name and description for the Package.

Create Package

Project Stream ZOSDEMO 1-0

Name Pack-001

Description Pack for working

Create Reset

Create Package

3. Once that is done, the User can select the required programs.

IKAN ALM will display all the files available for the selected Project Stream. The User can select the files and indicate the required revision number. Many types of files can be selected, built and deployed in the same process. (i.e., JCLs, Procs, Maps, Sysin, SQL, Sources with Assemble, COBOL, PL/1 languages, IDMS entities,..).

The following figure shows the package information before selection.

Package Info

Project Stream ZOSDEMO H_1-0

Package OID 15

Name Pack-001

Description Pack for working

Hidden No

Edit Refresh Overview

File Revisions Info

- Copy1.cpy
- ddl
- properties
 - ACCEPT.properties
 - ACCEPTX.properties
 - DEMO21.properties
 - ibmpli1s.properties
 - PROGRAM1.properties
 - SZSQL10.properties
- sql
- src
 - Accept.cbl
 - Acceptx.cbl.to_be_deleted
 - DEMO21.cbl
 - ibmpli1s.pli
 - Program1.cblicics
 - SZSQL10.cbl

Save Clear Refresh

Package info before selection

The next figure shows the package information after the selection of the required files.

Package info after selection

The final Package content would look as follows:

PATH	NAME	REVISION
/copy	Copy1.cpy	
/properties	DEMO21.properties	
/src	DEMO21.cbl	312

3 items found, displaying all

View package



Once the IKAN ALM User has defined the package, he can start building/compiling and, next, promoting or deploying his programs.

Step 3

Step 3: Select the required Action

To execute a build/compile or deploy, the User can simply click the required action button in the Action column.

DESKTOP Type	PROJECT STREAM (Package)	LEVEL	NEXT SCHEDULED	LATEST LEVEL REQUEST	LATEST SUCCESSFUL LEVEL REQUEST	ACTION	MESSAGE
Package	ZOSDEMO H_1-0 Pack-001	BUILDZOS		H 1-0 Pack-001 bi[16/10/13 22:21:28] ...	H 1-0 Pack-001 bi[16/10/13 22:21:28]		
	ZOSDEMO H_1-0 Pack-001	ZOSTEST		H 1-0 Pack-001 bi[16/10/13 23:20:48] ...	H 1-0 Pack-001 bi[16/10/13 23:20:48]		
	ZOSDEMO H_1-0 Pack-001	MILESTONE					No Build or Deploy Environments defined
	ZOSDEMO H_1-0 Pack-001	PRODZOS					

Step 4

Step 4: Create the Level Request

To start a compile for the project in our example, the User would click the appropriate Level Request action button at the Build Level. Next, the Create Level Request screen will be displayed.

Environments Info | Level: BUILDZOS | Package: Pack-001 | Project Stream: Head/1-0 | Project: ZOSDEMO

Build Environments					
NAME	MACHINE NAME	MACHINE OS	SOURCE LOCATION	TARGET LOCATION	BUILD SUFFIX
BUILDZOS	hercikanxp	WINDOWS	D:/ikan/ALM_environments/contbuild/source	D:/ikan/ALM_environments/contbuild/target	

Create Level Request : Request Build

Description

Indicative Build Number 1

Indicative VCR Tag

Define Build Parameters			
ENVIRONMENT	KEY	MACHINE PARAMETER	
BUILDZOS	ftp.password	*****	
BUILDZOS	ftp.tcpip	192.168.253.168	
BUILDZOS	ftp.userid	ADCDMST	
BUILDZOS	<input type="checkbox"/> dir.scripts	<input checked="" type="checkbox"/>	D:/ikan/ALM_system/ikanScripts
BUILDZOS	<input type="checkbox"/> script.location	<input checked="" type="checkbox"/>	D:/ikan/ALM_system/ikanScripts

[View package](#)

On this screen, the User can enter a description, view the parameters linked to the level request and, if configured that way, change some parameter values.

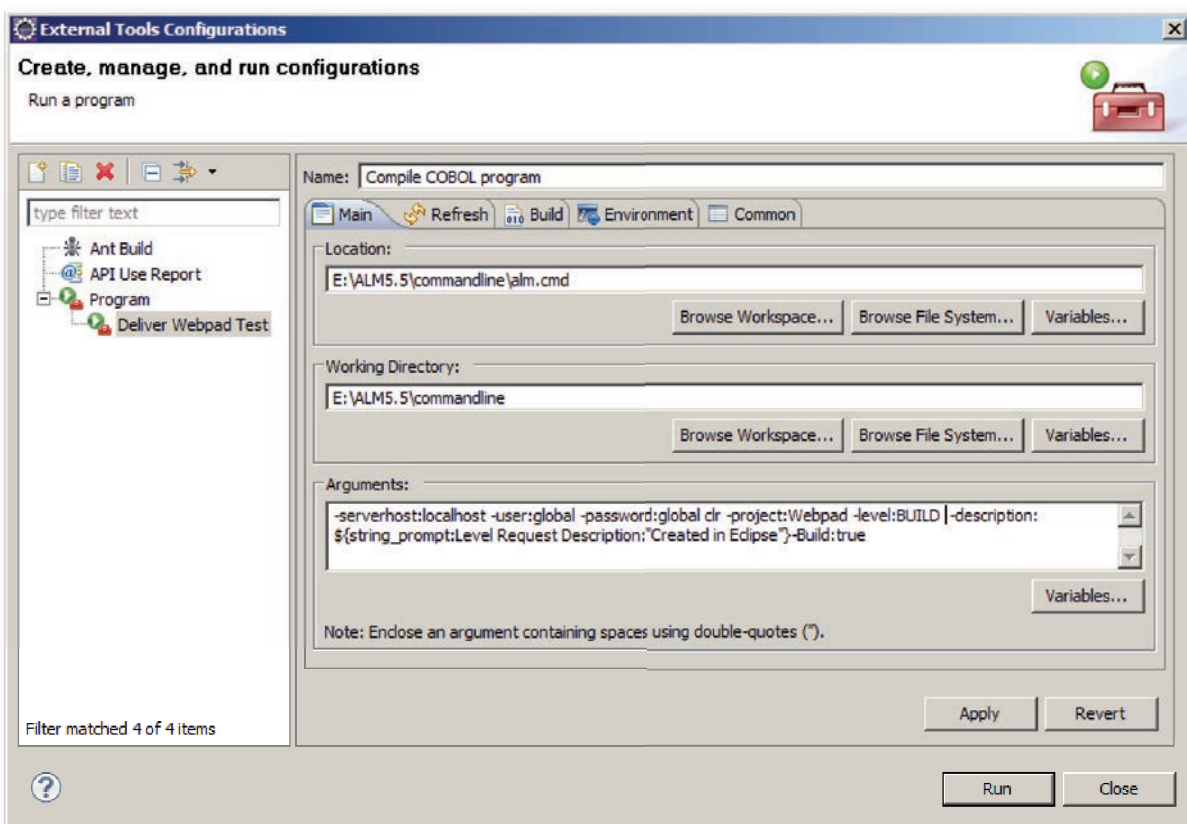
By clicking the Create button, the level request will be created and the process will start.



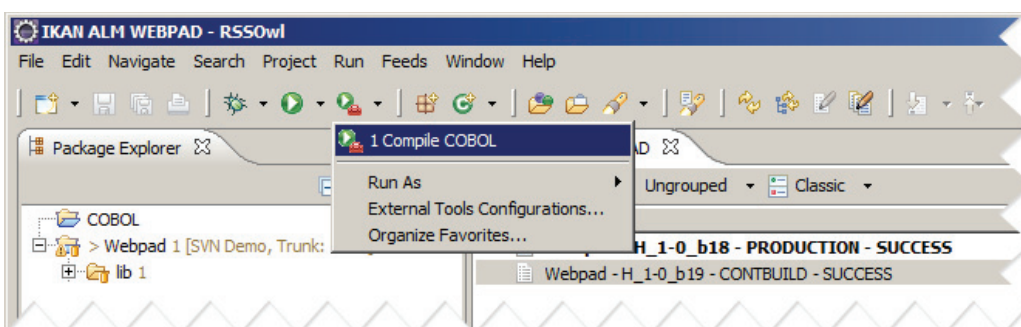
In fact, that is all a User needs to do for compiling all the programs in his package: go to his Desktop, click the appropriate action button and create a Level Request to execute a Build/Compile or a Deploy.

Alternative Way: Commandline Interface (CLI)

Another possibility to create a Compile/Build or Deploy action is to use the Commandline Interface (CLI) and to use the external tool configuration from Eclipse to configure the Level Request from within Eclipse.



External Tool Configuration from Eclipse



Launch IKAN ALM Level Request to Compile from within Eclipse

Additional information provided by IKAN ALM

Besides creating a Level request, IKAN ALM also provides a lot of additional information to a User:

- An overview of each Level Request.
- For each Level Request: detailed information through the interface.
- IKAN ALM reports. IKAN ALM comes with predefined reports, but the User can also define his own reports.

On the following pages we will show some sample IKAN ALM screens with more detailed information, such as the detailed Level Request information, the Build Log, the used Parameters, a sample report, a sample notification e-mail and the integration with an Issued Tracking System.

Finally, we will also show what happens behind the scene on the mainframe.

Level Request: 458	Project: ZOSDEMO	Project Stream: Head/1-0	Package: Pack-001	Level: BUILDZOS
------------------------------	----------------------------	------------------------------------	--------------------------	---------------------------

Type Builds based on latest code	VCR Tag H_1-0_Pack-001_b1
Action Type Request Build	Requested Date/Time 16/10/13 22:19:33
User ID Requester global	Start Date/Time 16/10/13 22:19:33
Description Compiling	End Date/Time 16/10/13 22:21:28
Status Warning	Duration 00:01:55

Logs	Level Parameters
-------------	-------------------------

PHASE	STATUS
Retrieve Code	Success
Build	Warning
Tag Code	Success
Deploy	Success
Issue Tracking	Success
Link File Revisions	Success
Cleanup Work Copy	Success

[View Sources](#) [View Issues](#)

Build Overview													
BUILD OID	BUILD TAGGED	BUILD NUMBER	BUILD STATUS	BUILD START DATE	BUILD END DATE	MACHINE NAME	BUILD ENVIRONMENT NAME	BUILD FILE NAME	VIEW CONTENT	ARCHIVE STATUS	FILE SIZE	PHASE DETAILS	FIRST ERROR PHASE
382		3	Warning	16/10/13 22:19:40	16/10/13 22:21:26	hercikanxp	BUILDZOS	ZOSDEMO_H_1-0_Pack-001_b1_BUILDZOS_#2.zip		Present	280 KB	Show Details	Cleanup Source

Related Issues				
ISSUE ID	DESCRIPTION	STATUS	OWNER	PRIORITY
Add Issue				

Level Request: Detailed Overview

Phase Log		
Phase z/OS programs Compilation		
Start Date/Time	2013-10-16 22:20:06.0	
End Date/Time	2013-10-16 22:21:24.0	
Duration	00:01:18	
Status	Success	
Message	Script Execution successful. Execution results in : D:\ikan\ALM_environments\contbuild\target\382	
Log		
Status SUCCESSFUL		
Total Time 1 minute 5 seconds		
Events		
TARGET	TASK	MESSAGE
callFilesProperties	[mkdir]	Created dir: D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
callFilesProperties	[delete]	Deleting directory D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
findSpecialProperties	[move]	Moving 1 file to D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\tempfiles
projectPropertiesError	[echo]	"WARNING: No D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\DemoMVS.properties file found for this project.
projectPropertiesError	[echo]	Default project properties are used.
startingConfiguration	[echo]	Level: BUILDZOS Config: ZOS - Action: Requested Build START
startingConfiguration	[delete]	Deleting: D:\ikan\ALM_environments\contbuild\source\382\DemoMVS\com.ikanalm.phases.ant.scripting.zosCompilation_2\alm_ant.properties.extended
execCompileZos	[mkdir]	Created dir: D:\ikan\ALM_environments\contbuild\target\382\jclwork
execCompileZos	[mkdir]	Created dir: D:\ikan\ALM_environments\contbuild\target\382\srcwork
execCompileZos	[delete]	Deleting directory D:\ikan\ALM_environments\contbuild\target\382\jclwork
execCompileZos	[delete]	Deleting directory D:\ikan\ALM_environments\contbuild\target\382\srcwork
execCompileZos	[mkdir]	Created dir: D:\ikan\ALM_environments\contbuild\target\382\jclwork
execCompileZos	[mkdir]	Created dir: D:\ikan\ALM_environments\contbuild\target\382\srcwork
getPropertyFile	[echo]	====> Properties found for DEMO21
setCompileProperties	[delete]	Deleting: D:\ikan\ALM_environments\contbuild\target\382\resources\DEMO21.properties
generateJclForCompilation	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\jclwork
addJclForCompileIbm	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\jclwork
addJclForCompileIbm	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\jclwork
addJclForCompileIbm	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\jclwork
addJclForCompileIbm	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\jclwork
addJclForCompileIbm	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\jclwork
addJclForCompileIbm	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\jclwork
generateJclForCompilation	[move]	Moving 1 file to D:\ikan\ALM_environments\contbuild\target\382
submitFileFTP	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382
submitFileFTP.active	[echo]	File D:\ikan\ALM_environments\contbuild\target\382\DEMO21.subftp was successful on ZOS.
getListingsFromCompile	[mkdir]	Created dir: D:\ikan\ALM_environments\contbuild\target\382\list
getListingsFromCompile	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\list
getLoadFile	[mkdir]	Created dir: D:\ikan\ALM_environments\contbuild\target\382\load
getLoadFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\contbuild\target\382\load
loadFilesFromCompile	[echo]	Generated Files are copied from ZOS.
loadFilesFromCompile	[delete]	Deleting: D:\ikan\ALM_environments\contbuild\target\382\DEMO21.jcl
execCompileZos	[delete]	Deleting directory D:\ikan\ALM_environments\contbuild\target\382\jclwork
execCompileZos	[delete]	Deleting directory D:\ikan\ALM_environments\contbuild\target\382\srcwork

Build log

View Build Log	Used Build Parameters
KEY	VALUE
alm.build.environmentName	BUILDZOS
alm.build.fileName	ZOSDEMO_H_1-0_Pack-001_b1_BUILDZOS_#2.zip
alm.build.machineName	hercikanxp
alm.build.number	3
alm.build.oid	382
alm.build.startDateTime	16-10-2013 22:19
alm.levelRequest.actionType	Requested Build
alm.levelRequest.buildType	Full Build
alm.levelRequest.levelName	BUILDZOS
alm.levelRequest.oid	458
alm.levelRequest.startDateTime	16-10-2013 22:19
alm.levelRequest.vcrTag	H_1-0_Pack-001_b1
alm.package.name	Pack-001
alm.project.buildToolTypeName	ANT
alm.project.deployToolTypeName	ANT
alm.project.description	Demo z/OS Project
alm.project.name	ZOSDEMO
alm.project.vcrName	SVNRepos
alm.project.vcrProjectName	DemoMVS
alm.projectStream.buildPrefix	1-0
alm.projectStream.type	H
ftp.password	*****
ftp.tcpip	192.168.253.168
ftp.userid	ADCDMST
source	D:\ikan\ALM_environments\contbuild\source\382\DemoMVS
sourceRoot	D:\ikan\ALM_environments\contbuild\source\382
target	D:\ikan\ALM_environments\contbuild\target\382

Used parameters

Project Streams Overview												
Description	Head	Status	Build Prefix	Locked	Tag-Based	Build Type	Partial Build VCR Tag	Accept Forced	VCR Branch ID	Build Suffix	Tag Template	
HEAD Project Stream for the packagestest2 Test Project	Head	Under construction	1-0	No	No	Full Build		No			`\${streamType}_\${prefix}_\${packageName}_b\${buildNumber}`	

Project Stream info	
Description	HEAD Project Stream for the packagestest2 Test Project
Life-Cycle	BASE
Build Prefix	1-0
Locked	No
Tag-Based	No
Highest Build Number	0
Accept Forced Build	No
VCR Branch ID	
Build Suffix	
Build Type	Full Build
Partial Build VCR Tag	
Status	Under construction

Project Stream Dependency

IKANALM Project Configuration Report: packagestest2 2 / 7

IKAN ALM Report sample, Project stream overview

Level Request for project DEMOZOS Base ended with status WARNING	
Project Information :	
Name :	DEMOZOS
Description :	Project DEMOZOS stored in Subversion for ZOS
VCR :	SVNRepository
VCR project :	DEMOZOS
Project Stream Information :	
Head :	yes
Build Prefix :	Base
Level Request Information :	
OID :	1352
Description :	test scripts
VCR Tag :	Base_b1
Created by :	Global Administrator
Created on :	2012-08-08 19:01:36.0
Sent to User Group :	ALM Project
Requested for :	2012-08-08 19:01:36.0
Action :	Request Build
Target Level :	BASE_BUILD
View LevelRequest Details	

Email received after successful Level Request

JIRA

Dashboards ▾ Projects ▾ Issues ▾

Webpad / WEB-1

Add the about menu

Edit Assign Comment More Actions ▾ Start Progress Resolve Issue Workflow ▾

Details

Type: New Feature Status: Open (View Workflow)

Priority: Major Resolution: Unresolved

Labels: None

Description

Add the about menu to the Webpad application

Activity

All Comments Work Log History Activity Source Reviews

global added a comment - 20/aug/13 5:31 PM

Related IKAN ALM Level Request : 21
<http://localhost:9080/alm/levelRequestAction.do?reqCode=displayDetailedOverview&oid=21&emailLink=true>
 Project : Webpad Level : CONTBUILD
 VCR Tag : H_1-0_b8 Build Number : 8

Integration with Issue Tracking: JIRA issue

What happens behind the scene?

The following z/OS screens show the corresponding actions on the mainframe.

The first z/OS Phase copies the copybook(s) and the programs (and, if existing, also the special components for compiling) to the z/OS environment.

The following screen shows the files that have been collected from the VCR and that are transferred via FTP to the mainframe in a PDS with IKAN ALM.DEMOS.TEST.SRCBATCH as PDS name.

```

BROWSE          IKANALM.DEMOS.TEST.SRCBATCH          Row 00001 of 00005
Command ==>                                         Scroll ==> CSR

```

Name	Prompt	Size	Created	Changed	ID
DEMO21		5732	2012/05/22	2013/10/17 03:19:40	ADCDMST
IBMPLI15		644	2012/05/25	2012/06/03 00:40:40	ADCDMST
SZSQL18		1985	2012/05/22	2013/09/21 01:34:40	ADCDMST
ULC018		5558	2013/04/29	2013/04/30 17:33:40	ADCDMST
YDP0188		1197	2001/03/29	2009/04/09 11:07:40	6552088

DEMO21 project

After this FTP Phase, the second phase, Z/OS program compile, is executed.

This phase creates the JCL (See Appendix V: Sample of z/OS compilation JCL), transfers that JCL via FTP to the mainframe and submits the job.

The results will also be available in the IKAN ALM Phase log. That log will list all executed events, step by step, and will tell if the compile has been executed successfully or not.

The following screen shows the JCL jobs that have been submitted and that are executed.

```

Display Filter View Print Options Help
-----
SDSF STATUS DISPLAY ALL CLASSES                LINE 1-28 (43)
COMMAND INPUT ==>                               SCROLL ==> CSR
PREFIX=* DEST=(ALL) OWNER=ADCDMST SYSNAME=
NP  JOBNAME JobID  Owner  Prty Queue  C Pos Max-RC  Saff ASys Status  PrtDest
-   ADCDMSTC JOB00135 ADCDMST  5 EXECUTION  A          15 JCL ERROR  LOCAL
    ADCDMSTC TSU00134 ADCDMST 15 EXECUTION          SYS1 SYS1      LOCAL
    ADCDMSTC JOB00001 ADCDMST  1 PRINT      A          15 JCL ERROR  LOCAL
    ADCDMSTC JOB00002 ADCDMST  1 PRINT      A          16 JCL ERROR  LOCAL
    
```

SDSF Status Display

The following screen shows the result of the execution on the mainframe:

```

SDSF OUTPUT DISPLAY ADCDMSTC JOB00135 DSID 2 LINE 0 COLUMNS 02- 133
COMMAND INPUT ==> _ SCROLL ==> CSR
***** TOP OF DATA *****
JES2 JOB LOG -- SYSTEM SYS1 -- NODE N1

21.28.26 JOB00135 ---- WEDNESDAY, 16 OCT 2013 ----
21.28.26 JOB00135 IRR010I USERID ADCDMST IS ASSIGNED TO THIS JOB.
21.28.28 JOB00135 ICH70001I ADCDMST LAST ACCESS AT 21:28:24 ON WEDNESDAY, OCTOBER 16, 2013
21.28.28 JOB00135 $HASP373 ADCDMSTC STARTED - INIT 1 - CLASS A - SYS SYS1
21.28.28 JOB00135 IEF403I ADCDMSTC - STARTED - TIME=21.28.28
21.28.29 JOB00135 -
--TIMINGS (MINS.)--
21.28.29 JOB00135 -STEPNAME PROCSTEP RC EXCP CONN TCB SRB CLOCK SERV WORKLOAD PAGE SWAP VIO SWAPS
21.28.29 JOB00135 -COPYSRC 00 93 0 .00 .00 .0 1441 BATCH 0 0 117 0
21.28.41 JOB00135 -COBOL 04 1630 0 .15 .00 .1 73427 BATCH 0 0 1189 0
21.28.41 JOB00135 -ALLOCLCT 00 15 0 .00 .00 .0 144 BATCH 0 0 2 0
21.28.41 JOB00135 -CREATLCT 00 61 0 .00 .00 .0 729 BATCH 0 0 3 0
21.28.41 JOB00135 -COPYLCT 04 58 0 .00 .00 .0 1894 BATCH 0 0 0 0
21.28.43 JOB00135 -LKEDT 04 263 0 .01 .00 .0 5718 BATCH 0 0 9 0
21.28.44 JOB00135 -CLEARSEQ 00 11 0 .00 .00 .0 107 BATCH 0 0 0 0
21.28.47 JOB00135 -XMITLOAD 00 655 0 .03 .00 .0 16543 BATCH 0 0 0 0
21.28.48 JOB00135 -PRNTCOMP 00 396 0 .01 .00 .0 7209 BATCH 0 0 0 0
21.28.48 JOB00135 $HASP375 ADCDMSTC ESTIMATED LINES EXCEEDED
21.28.48 JOB00135 -PRNTLINK 00 61 0 .00 .00 .0 775 BATCH 0 0 0 0
21.28.49 JOB00135 -FRMTLKD 00 45 0 .00 .00 .0 1174 BATCH 0 0 0 0
    
```

SDSF Output Display

When the Job is completed with success, the listing and the load module are transferred via FTP to the IKAN ALM target environment. The following screen shows the sequential file generated by the Xmit step for transferring the load module to the IKAN ALM target environment.

```

BROWSE      IKANALM.DEMOS.TEST.DEMO21                Line 00000000 Col 001 000
Command ==>                                         Scroll ==> CSR
***** Top of Data *****
\INMR01.B...&.....NODENAME.....ADCDMST.....ADCD.....*.....20131017032044.
.....=INMR02.....IEBCOPY.....1%.....~.....B.....;I..
.{.....IKANALM.DEMOS.TEST.LOADLIB.....-..DEMO21\INMR02.....
INMCOPI.....1%.....B....."4.....I.....H.....\INMR03.....1%.....
&.....I.....{.....~.....;.....{....."8.....Vs.....K.....^.....~.....>D.....~.....
..Q.....↑.qQ5.....L.....0.....Q5.....φ.....2.....Q5.....".....b.....Q5.....l.....j.....
.....
.....~.....DEMO21.....BS.
H.....h.....
.....~.....L.....h.....~.....~DEMO21.....4HIGZCBSO.....CEESTART.....CEEBE
BL.....CEERAN0.....CEELOCT.....CEEGMT0.....CEESG005.....
.....~3

```

Sequential Xmit file

As a result, IKAN ALM has the listing and the load module in his archive. At this point, IKAN ALM is in full control of the remainder of the steps in the lifecycle.



Content of Build Archive

From the IKAN ALM archive, the load module can be deployed or promoted to a test or production level by simple starting a Deploy (promote) Level Request that executes a receive step.

The next step is the Deploy or Promote of the compile results. The following screen starts the Deploy.

The screenshot displays the IKAN ALM interface for creating a level request. At the top, there are navigation tabs: 'Environments Info', 'Level: ZOSTEST', 'Package: Pack-001', 'Project Stream: Head/1-0', and 'Project: ZOSDEMO'. Below this is a 'Deploy Environments' table:

#	NAME	MACHINE NAME	MACHINE OS	SOURCE LOCATION	TARGET LOCATION
0	ZOSTEST	hercikanxp	WINDOWS	D:/ikan/ALM_environments/testdeploy/source	D:/ikan/ALM_environments/testdeploy/target/ZOSDEMO

Below the table is the 'Create Level Request : Deliver Build' section. It includes a 'Description' field with the value 'deploy demo021' and a 'Requested Date/Time' field. Underneath is a 'Select Build to deliver' table:

BUILD NUMBER	LEVEL REQUEST OID	DESCRIPTION	LEVEL REQUEST VCR TAG	LEVEL REQUEST END	AVAILABLE ON LEVEL	ACTIVE BUILD
3	458	Compiling	H_1-0_Pack-001_b1	16/10/13 22:21:28	BUILDZOS	<input checked="" type="checkbox"/>

Buttons for 'Create', 'Reset', and 'Back' are located below the table. The bottom section is 'Define Deploy Parameters', which is a table with columns for 'ENVIRONMENT', 'KEY', and 'MACHINE PARAMETER':

ENVIRONMENT	KEY	MACHINE PARAMETER
ZOSTEST	dir.zosProperties	D:/ikan/ALM_system/phaseProps
ZOSTEST	ftp.password	****
ZOSTEST	ftp.tcpip	192.168.253.168
ZOSTEST	ftp.userid	ADCDMST
ZOSTEST	<input checked="" type="checkbox"/> env.qualifA	TEST
ZOSTEST	<input checked="" type="checkbox"/> dir.scripts	<input checked="" type="checkbox"/> D:/ikan/ALM_system/ikanScripts
ZOSTEST	<input checked="" type="checkbox"/> script.location	<input checked="" type="checkbox"/> D:/ikan/ALM_system/ikanScripts

Starting a deploy

For the Deploy, IKAN ALM is using the same process as for a Compile/Build: **A Deploy Level and Environment with its related z/OS phases must be created.**

The z/OS phases defined here are:

- The promote (FTP) of load-modules and other components to the mainframe
- Delete obsolete files and associated components (such as load modules, listings)
- The DB2 Bind statements, transfer (FTP) and DB2 Job execution
- The activation of the CICS load-modules

During this Level Request, the files are copied via FTP to their respective PDS(s) and special jobs can be created and executed on z/OS. The Load-modules are received through a transmitted sequential file to the PDS.

The FTP and Job results are analyzed for validating the executed deployment actions. In case of errors, messages are transmitted to the IKAN ALM log and the deployment is stopped.

IKAN ALM log

Phase Log | **Phase Parameters**

Phase Promote components and load-modules to z/OS
Start Date/Time 2013-10-16 23:19:57.0
End Date/Time 2013-10-16 23:20:29.0
Duration 00:00:32
Status Success
Message Script Execution successful.
 Execution results in : D:\ikan\ALM_environments\testdeploy\target\ZOSDEM0

Log
Status SUCCESSFUL
Total Time 23 seconds

Events

Task	Type	Message
callFilesProperties	[mkdir]	Created dir: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_2\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_3\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_3\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_3\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_3\templates
callFilesProperties	[delete]	Deleting directory D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_2\templates
findSpecialProperties	[move]	Moving 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_2
projectPropertiesError	[echo]	**WARNING**: No D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_2\properties file found for this project.
projectPropertiesError	[echo]	Default project properties are used.
startingConfiguration	[echo]	Level: ZOSTEST Config: ZOS - Action: Deliver Build START
startingConfiguration	[delete]	Deleting: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosPromotion_2\vm_and_properties_extended
initPromoteZos	[delete]	Deleting directory D:\ikan\ALM_environments\testdeploy\target\ZOSDEM0
copyObjectsZos	[copy]	Copying 2 files to D:\ikan\ALM_environments\testdeploy\target\ZOSDEM0
copyFileToZos	[echo]	Selected files are copied to z/OS
submitFileFTP	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278
submitFileFTP.active	[echo]	File Pspfile.smbrpt was successful on z/OS
zostsarexloads	[echo]	Selected zos-modules are loaded to z/OS z/OS

Phase Log | **Phase Parameters**

Phase z/OS Delete Sources and associated objects
Start Date/Time 2013-10-16 23:20:30.0
End Date/Time 2013-10-16 23:20:37.0
Duration 00:00:07
Status Success
Message Script Execution successful.
 Execution results in : D:\ikan\ALM_environments\testdeploy\target\ZOSDEM0

Log
Status SUCCESSFUL
Total Time 3 seconds

Events

Task	Type	Message
callFilesProperties	[mkdir]	Created dir: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3\templates
callFilesProperties	[delete]	Deleting directory D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3
findSpecialProperties	[move]	Moving 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3
projectPropertiesError	[echo]	**WARNING**: No D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3\properties file found for this project.
projectPropertiesError	[echo]	Default project properties are used.
startingConfiguration	[echo]	Level: ZOSTEST Config: ZOS - Action: Deliver Build START
startingConfiguration	[delete]	Deleting: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosDeleteObsoleteFiles_3\vm_and_properties_extended
deleteObsoleteFiles	[echo]	Level: ZOSTEST Config: ZOS - Action: Deliver Build for deleting files DONE

Phase Log | **Phase Parameters**

Phase z/OS DB2 Blnds transfer and activation
Start Date/Time 2013-10-16 23:20:37.0
End Date/Time 2013-10-16 23:20:42.0
Duration 00:00:05
Status Success
Message Script Execution successful.
 Execution results in : D:\ikan\ALM_environments\testdeploy\target\ZOSDEM0

Log
Status SUCCESSFUL
Total Time 1 second

Events

Task	Type	Message
callFilesProperties	[mkdir]	Created dir: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\templates
callFilesProperties	[delete]	Deleting directory D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\templates
findSpecialProperties	[move]	Moving 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4
projectPropertiesError	[echo]	**WARNING**: No D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\properties file found for this project.
projectPropertiesError	[echo]	Default project properties are used.
startingConfiguration	[echo]	Level: ZOSTEST Config: ZOS - Action: Deliver Build START
startingConfiguration	[delete]	Deleting: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosBindDb2_4\vm_and_properties_extended
ignoreDb2zos	[echo]	*** No bind submitted to DB2/DB2 ***

Phase Log | **Phase Parameters**

Phase z/OS Cics Load-modules activation
Start Date/Time 2013-10-16 23:20:43.0
End Date/Time 2013-10-16 23:20:47.0
Duration 00:00:04
Status Success
Message Script Execution successful.
 Execution results in : D:\ikan\ALM_environments\testdeploy\target\ZOSDEM0

Log
Status SUCCESSFUL
Total Time 1 second

Events

Task	Type	Message
callFilesProperties	[mkdir]	Created dir: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\templates
selectPropertyFile	[copy]	Copying 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\templates
callFilesProperties	[delete]	Deleting directory D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\templates
findSpecialProperties	[move]	Moving 1 file to D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5
projectPropertiesError	[echo]	**WARNING**: No D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\properties file found for this project.
projectPropertiesError	[echo]	Default project properties are used.
startingConfiguration	[echo]	Level: ZOSTEST Config: ZOS - Action: Deliver Build START
startingConfiguration	[delete]	Deleting: D:\ikan\ALM_environments\testdeploy\source\278\com.ikanalm.phases.ant.scripting.zosUpdateCics_5\vm_and_properties_extended
runUpdateCics	[echo]	*** Environment CICS is active ***
runUpdateCics	[mkdir]	Created dir: D:\ikan\ALM_environments\testdeploy\source\278\temp\cics
ignoreCicsZos	[echo]	*** No Cics selected submitted to CICS z/OS ***

The following screen shows the JCL for receiving this transmitted sequential file.

```
//ADCDMSTR JOB (5145,00000,2233,T),'IKAN',
//      MSGLEVEL=(1,1),MSGCLASS=X,
//      CLASS=A,REGION=0M
//*      NOTIFY=&SYSUID,TYPRUN=SCAN
//**REQ ROUTEID=ADCD
//*****
//** RECEIVE PROGRAM **
//*****
//RECVLOAD EXEC PGM=IKJEFT01,COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//* RECEIVE INDS('recvfile')
//* DA('loadlib') SHR
//SYSTSIN DD *
RECEIVE INDSNAME('IKANALM.DEMOS.TEST.DEMO21.SEQ')
DA('IKANALM.DEMOS.TEST.LOADLIB') SHR
/*
//*-----
//*---      DELETE RECEIVED FILES
//*-----
//CARECV EXEC PGM=IEFB14
//DD01 DD DISP=(MOD,DELETE,DELETE),
//      DSN=IKANALM.DEMOS.TEST.DEMO21.SEQ,
```

Receive program

The following screen shows the load modules after FTP from the IKAN ALM archive to the mainframe in a PDS:

```
Menu Utilities Compilers Help
BROWSE IKANALM.DEMOS.TEST.LOADLIB(DEMO21) Line 00000000 Col 001 132
Command ==> Scroll ==> CSR
***** Top of Data *****
~.....~DEMO21 .....4HIGZCISO ..... CEESTART..... CEEBETBL..... CEERANO ..... CEELCOT ..... CEEGMO ..... CEESG005...
.....
..5695PMB01 .....+....
..~.5655G5300 .....+
..h.....+.DEMO21/ADCDMSTR/000001
.....
000.CEE...q...q00.q01...x.....q00}.0.0.q00<.....\H...z.....6.....~.....DEMO21 20131016212029030401.E...%
.....0.....D...H...<...M...F...~...0...*...-...U...Y...%...0...E...<...@...~...d...h...<...q...e...
.....0...k..._q.....<...D...H...<...8...M...q...*...-...U...Y...%...0...E...<...@...~...d...h...<...q...e...n
U@.K.(Y@.K.(%@.K.(@@.K.(~@.K.(d@.k?{hK.(q@.K.(k@.K.(m@.K.(a@.k (AK@(+AK.+{QK.+u{QK.+w{QK.+y{QK.+n{QK.+l{QkY+}K.
.....q-.I...I.
H.~|l...q-j.q |.l ...B.LQ@{G{b! {.) {.& |.H.~|l...Q@j.q @2l ...B.LG{+n! {.) {.& @2E\jQ@{G0Z.E\j@E\jQ.j.H..0@ j.<@E\jQ.j.q.j.
.....3~_q._q
[. .FN..@.Q@{m@}+0@.Jq.3E\j@E\jQ.j.q.{UK..V..E\jQ.j.q.{UK..U..E\jQ.j.q.{UK..Y..E\jQ.j.q.{UK..0..E\jQ.j.q.{U
-q..._q.....<...M...-...U...Y...a...+...H...&...Q..._q
k...x8...~...y...~...2.yb...~...B.y@...~...B.zs...~...B.zF...~...K.nk...~...K.nx...~...s.nK...~...v.
.....0..|.
.....M...$......;q.^r.;q..IA...B.....M...s.....-Z
.._q.._q
.....M...:.....In.n].p@...J.....~.....M...'......IZ.n]
.....vh .x-.x-
```

PDS with load modules

At this moment our programs are available for testing in the z/OS Test environment.

The deployment to another z/OS Environment, be it another LPAR or Production environment, is a similar process.

IKAN ALM –Administrator’s Point of View



Before Users can start working in IKAN ALM, the IKAN ALM Administrator needs to set up and configure IKAN ALM. Next, he will take care of creating the IKAN ALM projects and specifying the required parameters for the environments and phases.

To make his task as easy as possible, IKAN ALM has introduced the concept of Phases. Phases allow the IKAN ALM Administrator to customize the workflow of the projects by using highly reusable building blocks. Phases can be shared between different Projects, but also between different IKAN ALM installations.

He will use and customize the IKAN ALM pre-defined “Core” phases to transfer the required components to the main-frame, to create the necessary JCL, to submit the JCL and to transfer the results back to IKAN ALM. If needed, he can also create his own “Custom” Phases.

Step 1





Step1: Create the global Phases

The Phase concept









To compile or deploy programs one or more phases are executed via IKAN ALM.

In this section, we will first explain the different components of the phases and, next, we will show how a phase is represented and used in IKAN ALM.

The IKAN ALM Phase Structure

 common	20/09/2013 11:50
 models	23/09/2013 18:38
 resources	21/08/2013 10:02
 zosCompilation.xml	08/10/2013 10:06

The Common Files

 getZosProgramProperties.xml	08/10/2013 14:47
 initZosInfos.xml	08/10/2013 14:47
 jchck000.exe	08/10/2013 14:47
 jchck010.exe	08/10/2013 14:47
 linkEditSyslin.xml	08/10/2013 14:47
 loadProperties.xml	08/10/2013 14:47
 specialProps.xml	08/10/2013 14:47
 zosActionsFTP.xml	08/10/2013 14:47

As an example, we have here a common script file that is used for linking a COBOL program. This common file will be used as a template to finally generate the correct and complete JCL step for linking a program.

```








<?xml version="1.0" encoding="UTF-8"?>
<!-- Standard options for linkedit SYSLIN parameters -->
- <project basedir="." default="linkSyslin" name="linkEditSyslin">
  <!-- ***** -->
  <!-- Completing SYSLIN cards depending on object type and PGM options-->
  <!-- ***** -->
  - <target name="linkSyslin">
    <echo file="{syslinFile}"> // DD * </echo>
    - <if>
      <isset property="include.syslib1"/>
      - <then>
        <echo file="{syslinFile}" append="true"> {include.syslib1}</echo>
      </then>
    </if>
    - <if>
      <isset property="include.syslib2"/>
      - <then>
        <echo file="{syslinFile}" append="true"> {include.syslib2}</echo>
      </then>
    </if>
    - <if>
      <isset property="include.syslib3"/>
      - <then>
        <echo file="{syslinFile}" append="true"> {include.syslib3}</echo>
      </then>
    </if>
    <echo file="{syslinFile}" append="true"> /* // DD DSN=&&OBJECT,DISP=(OLD,DELETE)</echo>
    - <if>
      <isset property="pgm.datacom"/>
      - <then>
        <echo file="{syslinFile}" append="true"> /* DD DSN=DATACOM.DEFAULTS(MEMBER),DISP=SHR </echo>
      </then>
    </if>
    <echo file="{syslinFile}" append="true"> // DD DSN=&&LCTFILE({member}),DISP=(OLD,DELETE) </echo>
    - <if>
      <istrue value="{includeName}"/>
      <!-- it's the standard case -->
      - <then>
        <echo file="{syslinFile}" append="true"> // DD * IDENTIFY {pgm.loadname}('{package.zos.identify}')</echo>
      </then>
      <!-- it's the PLI case -->
      - <else>
        <echo file="{syslinFile}" append="true"> IDENTIFY CEESTART('{package.zos.identify}')</echo>
      </else>
    </if>
    <echo file="{syslinFile}" append="true"> NAME {pgm.loadname}(R) /* </echo>
  </target>
</project>

```

[Link file template](#)

The Resource Files

Resource files are used to define specific, reusable properties

 configuration.properties	08/10/2013 14:47
 defaultPgms.properties	08/10/2013 14:47
 environment.properties	08/10/2013 14:47
 extensions.properties	08/10/2013 14:47
 languages.properties	08/10/2013 14:47
 osfamily.properties	08/10/2013 14:47
 parmsFTP.properties	08/10/2013 14:47

As an example, the COBOL2 language definitions from the languages.properties file:

```
# -----
# Properties for ZOS Languages
# - called after (Environment).properties
# - property format: (language).parameter
# -----
# env.prefix=
# env.qualifA=
# COBOL2 compilation parameters
COBOL2.program=IGYCRCTL
COBOL2.parms=LIST,LIB,NOSEQ,NOCMPR2,MAP
COBOL2.parmlib=${env.prefix}.${env.qualifA}.PARMLIB
COBOL2.prefix=SYS1.CEE
COBOL2.loadlib=${COBOL2.prefix}.SIGYCOMP
COBOL2.suffix=SCEELKED
COBOL2.link.parms=LIST,MAP,XREF,NCAL
COBOL2.cics.program=DFHECP1$
COBOL2.cics.parms=COBOL2,LANGLVL(2),NODEBUG,NOSOURCE,SP,NOOPT
COBOL2.cics.linkModule=DFHECI
COBOL2.cics.db2.linkModule=DSNCLI
COBOL2.db2.program=DSNHPC
COBOL2.db2.parms=HOST(COB2),APOST
COBOL2.db2.linkModule=DSNCLI
COBOL2.dtcn.program=DBXMMPR
COBOL2.dtcn.parms=DBOPTBAC
COBOL2.dtcn.parms.cics=DBOPTCIC
COBOL2.dtcn.linkModule=DBXHVPR
COBOL2.idms.program=IDMSC
COBOL2.idms.parms=(COBOL)
COBOL2.idms.linkModule=IDMSCBL
COBOL2.ims.program=DSNIIPC
COBOL2.ims.parms=(COBOL2)
COBOL2.ims.linkModule=DSNILI
COBOL2.linkedit.program=HEWL
COBOL2.ndvr.type=COBOL
```

The example below shows the generated properties to use the COBOL2 language definitions in a COBOL program:

























```
#Ant properties
#Mon Sep 30 10:57:09 CEST 2013
pgm.amode=31
pgm.cics=true
pgm.compile.parms=DATA(31),${RENT}
pgm.compilerType=IBM
pgm.db2=true
```

```
pgm.db2.collection=xxxxIKAN1
pgm.db2.path=xxxxFUNC
pgm.db2.plan=ULC010
pgm.db2.sql=true
pgm.debugger=true
pgm.language=COBOL2
pgm.link.parms=
pgm.loadname=ULC010
pgm.name=ULC010
pgm.noname=true
pgm.os=ZOS
pgm.reus=RENT
pgm.rmode=ANY
pgm.type=program
```

The Model files

Model files are used as templates for JCL steps.

As an example, we added a model for a JCL to compile a COBOL program, with a COBOL2 compiler.

 addEndevor_jcl.model	08/10/2013 14:47
 bindpkg_sysin.model	08/10/2013 14:47
 bindplan_sysin.model	08/10/2013 14:47
 compileAsm_jcl.model	08/10/2013 14:47
 compileBms_jcl.model	08/10/2013 14:47
 compileCobol_jcl.model	08/10/2013 14:47
 compileEndevor_jcl.model	08/10/2013 14:47
 compilePli_jcl.model	08/10/2013 14:47
 copyLct_jcl.model	08/10/2013 14:47
 copySource_jcl.model	08/10/2013 14:47
 debugXpediter_jcl.model	08/10/2013 14:47
 jobCard_jcl.model	08/10/2013 14:47
 linkEdit_jcl.model	08/10/2013 14:47
 listing_jcl.model	08/10/2013 14:47
 precompileCICS_jcl.model	08/10/2013 14:47
 precompileDatacomCobol_jcl.model	08/10/2013 14:47
 precompileDatacomExportImport_jcl.mo...	08/10/2013 14:47
 precompileDatacomPli_jcl.model	08/10/2013 14:47
 precompileDb2_jcl.model	08/10/2013 14:47
 precompileDms_jcl.model	08/10/2013 14:47
 project_properties.model	08/10/2013 14:47
 sample_file.model	08/10/2013 14:47
 Submit_cmd.model	08/10/2013 14:47
 xmitload_jcl.model	08/10/2013 14:47

The compileCobol_jcl.model

```
/**
*****
**  COMPILE COBOL2, store object in objlib if compile=ok
**  compile listing is stored in ${env.lib.listA}
*****
//      SET PARMCOB='${lang.parms}'
//      SET PARMCOB0='${pgm.compile.parms}'
*****
**  COMPILE THE ELEMENT                                **
*****
/${pgm.language} EXEC PGM=${lang.program},COND=(4,LT),
//  PARM='&PARMCOB0,&PARMCOB',MAXRC=4
//STEPLIB DD DISP=SHR,DSN=${lang.loadlib}
//SYSIN DD DISP=(OLD,PASS),DSN=&&SRCOMPIL
//SYSLIN DD DISP=(,PASS),DSN=&&OBJECT,
//      UNIT=SYSDA,SPACE=(CYL,(2,2)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT7 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSPRINT DD DISP=(,PASS),DSN=&&COMPLIST,
//      UNIT=SYSDA,SPACE=(TRK,(10,10),RLSE)
/**      DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
/**
//SYSLIB DD DISP=SHR,DSN=${copylib}
/${copylib1}
/${copylib2}
/${copylib3}
/${copylib4}
/${inc2lib0}
/${inc2lib1}
/${inc2lib2}
/${inc2lib3}
/${inc2lib4}
```

JCL model for COBOL compile step

An IKAN ALM phase and its usage: the z/OS compile phase

The figure below shows the z/OS compile phase.

The screenshot displays the 'Edit Phase' configuration for the 'z/OS programs Compilation' phase. The configuration includes the following details:

- Name:** com.ikanalm.phases.ant.scripting.zosCompilation
- Version:** 1.0.0
- Default Display Name:** z/OS programs Compilation
- Display Name [English]:** z/OS programs Compilation
- Display Name [French]:** Compilation des programmes z/OS
- Display Name [German]:** z/OS programs Compilation
- Description:** z/OS all languages programs Compilation. This phase works after the zosCopyForCompilation phase only! May be customized with property files and model files.
- Author:** IKAN
- Execution Type:** ANT
- Core Phase:** No
- Certified:** No
- Released:** No
- Uploaded Files:** A list of files including zosCompilation.xml, ant-contrib-1.0.03.jar, ant-ikan-tools.jar, and others.
- Phase can be used on:**
 - Level:** Yes No
 - Build Environment:** Yes No
 - Deploy Environment:** Yes No

Below the configuration is a table of 'Phase Parameters' with 11 items found, displaying all:

Name	Default Value	Description	Mandatory	Secure	Integration Type
propsfile.languages	./resources/languageOptions.properties	property File of z/OS languages			None
propsfile.extensions	./resources/extensions.properties	property File of z/OS extensions and directories in LUW environment			None
propsfile.parmsFTP	./resources/parmsFTP.properties	property File of z/OS FTP system			None
propsfile.parmsZOS	./resources/parmsZOS.properties	property File of z/OS environment			None
ftp.active	true	Option for transmitting to z/OS (true or false)			None
ftp.password	*****	Password of FTP userid		<input checked="" type="checkbox"/>	None
dir.zosProperties	./resources	Property files Location			None
dir.zosModels	./models	Models Location			None
alm.phase.builder			<input checked="" type="checkbox"/>		ANT
alm.phase.extractBundle	true		<input checked="" type="checkbox"/>		None
alm.phase.mainScript	zosCompilation.xml	All program types Compilation	<input checked="" type="checkbox"/>		None

The objective of the z/OS compile Phase is to compile z/OS programs with, mainly, Assembler, COBOL and PL/1, BMS map languages, and working with CICS and Databases as DB2, Datacom, IDMS or IMS. The Phase will also control the results of the JCL submit and it will collect all files generated by the compile Jobs. Also, when applicable, the DB2 Bind Files will be generated.

This phase assumes that the files for compiling sources and the source program files have already been transferred to the mainframe in the correct PDSs. Normally, this would be done by a dedicated phase.

It is the task of the IKAN ALM Administrator to make sure that the default values for the parameters are set to the company standards. He can easily do that by changing the parameter values in the IKAN ALM web interface.

Phase Parameters						
Name	Default Value	Description	Mandatory	Secure	Integration Type	
propsfile.languages	./resources/languageOptions.properties	property File of z/OS languages			None	
propsfile.extensions	./resources/extensions.properties	property File of z/OS extensions and directories in LUW environment			None	
propsfile.parmsFTP	./resources/parmsFTP.properties	property File of z/OS FTP system			None	
propsfile.parmsZOS	./resources/parmsZOS.properties	property File of z/OS environment			None	
ftp.active	true	Option for transmitting to z/OS (true or false)			None	
ftp.password	*****	Password of FTP userId		✓	None	
dir.zosProperties	./resources	Property files Location			None	
dir.zosModels	./models	Models Location			None	
alm.phase.builder			✓		ANT	
alm.phase.extractBundle	true		✓		None	
alm.phase.mainScript	zosCompilation.xml	All program types Compilation	✓		None	

11 items found, displaying all

The execution of the z/OS Compile phase will use the z/OS compile script to finally generate a complete JCL, taking into account all JCL steps to be executed.

The figure below shows the generated JOB card and the STEP card to compile the COBOL program. The complete generated JCL can be found in Appendix V: Sample of z/OS compilation JCL.

```

...
//*****
//**   COMPILE COBOL2, store object in objlib if compile=ok
//**   compile listing is stored in IKAN ALM.DEMOS.TEST.LSTALIB
//*****
//       SET PARMCOB='LIST,LIB,NOSEQ,NOCMR2,MAP'
//       SET PARMCOB0='DATA(31)'
//*****
//**   COMPILE THE ELEMENT                                     **
//*****
//COBOL2 EXEC PGM=IGYCRCTL,COND=(4,LT),
//   PARM='&PARMCOB0,&PARMCOB'
//STEPLIB DD DISP=SHR,DSN=SYS1.COB2COMP
//SYSIN   DD DISP=(OLD,PASS),DSN=*&&SRCOMPIL          (ULC010)
//SYSLIN  DD DISP=(,PASS),DSN=*&&OBJECT,
//         UNIT=SYSDA,SPACE=(CYL,(2,2)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSUT1  DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT2  DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT3  DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT4  DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT5  DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT6  DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT7  DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSPRINT DD DISP=(,PASS),DSN=*&&COMPLIST,
//         UNIT=SYSDA,SPACE=(TRK,(10,10),RLSE)
//*       DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
//*
//SYSLIB  DD DISP=SHR,DSN=IKAN ALM.DEMOS.TEST.COPYLIB
//         DD DISP=SHR,DSN=IKAN ALM.DEMOS.INTG.COPYLIB
//         DD DISP=SHR,DSN=IKAN ALM.DEMOS.QUAL.COPYLIB
//         DD DISP=SHR,DSN=IKAN ALM.DEMOS.PROD.COPYLIB
//*

```

The IKAN ALM Administrator will do this for all the phases required for the mainframe Build (compile) and deploy processes (probably for several z/OS projects).

An IKAN ALM phase and its usage: the z/OS deployment phase

Next, we will show you an example of a Deployment Phase.

The main phase of the z/OS deployment with IKAN ALM is the z/OS Promotion of components and load-modules' Phase.

Edit Phase

Name	com.ikanalm.phases.ant.scripting.zosPromotion		
Version	1.0.0		
Default Display Name	z/OS Promotion of components and load-mo		
Display Name [English]	Promote components and load-modules to z		
Display Name [French]	Promotion de composants et load-modules		
Display Name [German]	Promote components and load-modules to z		
Description	Promote components and load-modules to z/OS environment. May be customized with property files and model files.		
Author	IKAN		
Execution Type	ANT		
Core Phase	No		
Certified	No		
Released	No		
Uploaded Files	antext/lib/xmltask.jar common/copySourceToTarget.xml common/initZosInfos.xml common/jchck000.exe common/jchck010.exe		Upload
Phase can be used on:			
Level	<input type="radio"/> Yes <input checked="" type="radio"/> No		
Build Environment	<input type="radio"/> Yes <input checked="" type="radio"/> No		
Deploy Environment	<input checked="" type="radio"/> Yes <input type="radio"/> No		

[History](#)

[Save](#) [Refresh](#) [Overview](#)
[Release](#) [Export](#) [Copy](#)

Phase Parameters						
	NAME	DEFAULT VALUE	DESCRIPTION	MANDATORY	SECURE	INTEGRATION TYPE
	propsfile.extensions	./resources/extensions.properties	property File of z/OS extensions and directories in LUW environment			None
	propsfile.parmsFTP	./resources/parmsFTP.properties	property File of z/OS FTP system			None
	propsfile.environment	./resources/environment.properties	property File of z/OS environment			None
	ftp.active	true	Option for transmitting to z/OS (true or false)			None
	ftp.password	*****	Password of FTP userId			None
	dir.zosProperties	./resources	Property files Location			None
	dir.zosModels	./models	Models Location			None
	alm.phase.builder					ANT
	alm.phase.extractBundle	true				None
	alm.phase.mainScript	zosPromotion.xml	Promote components and load-modules to z/OS environment			None

10 items found, displaying all

[Create Parameter](#)

Typically, the parameters are very similar, but you can specify another property file for the target environment during the project setup if required.

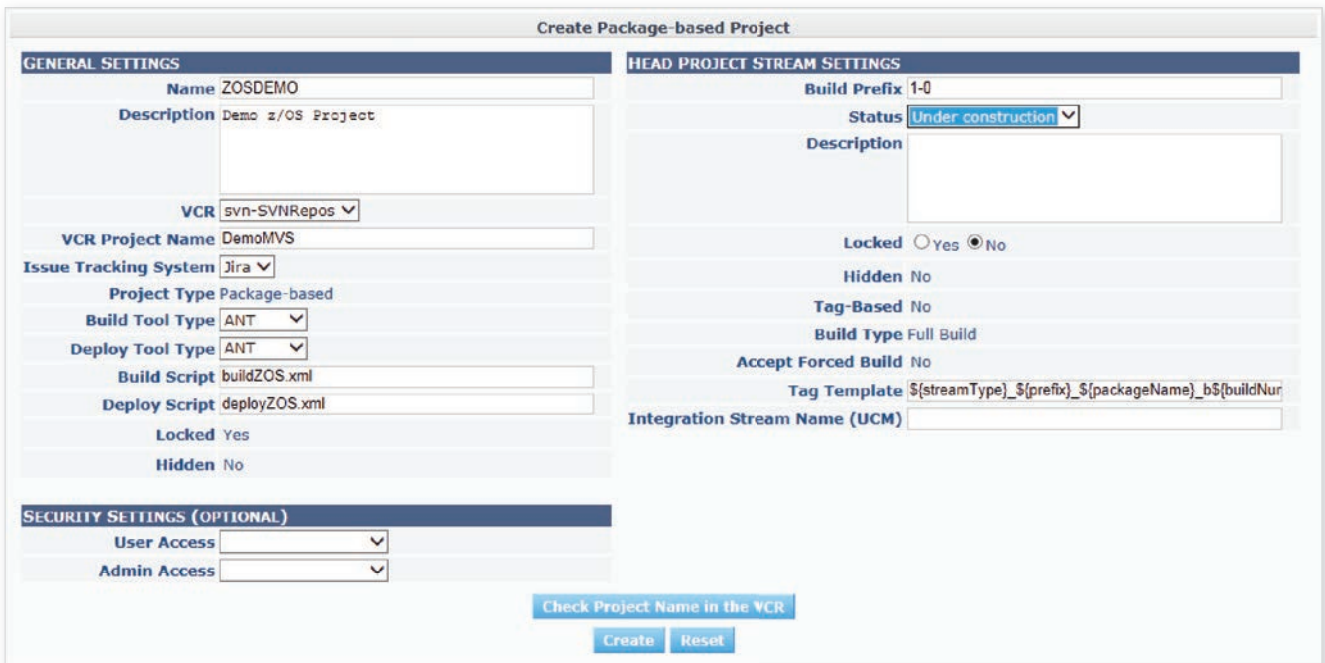
Step 2

Step 2: Create the IKAN ALM project(s)

Once the global phases have been defined, the IKAN ALM Administrator creates a release-based or package-based mainframe project.

In this example we will show the creation of a package-based mainframe project: ZOSDEMO.

First he needs to define the General settings and the Project Stream (Baseline) settings on the following screen.



GENERAL SETTINGS	HEAD PROJECT STREAM SETTINGS
Name: ZOSDEMO	Build Prefix: 1-0
Description: Demo z/OS Project	Status: Under construction
VCR: svn-SVNRepos	Description:
VCR Project Name: DemoMVS	Locked: <input type="radio"/> Yes <input checked="" type="radio"/> No
Issue Tracking System: Jira	Hidden: No
Project Type: Package-based	Tag-Based: No
Build Tool Type: ANT	Build Type: Full Build
Deploy Tool Type: ANT	Accept Forced Build: No
Build Script: buildZOS.xml	Tag Template: \${streamType}_\${prefix}_\${packageName}_b\${buildNur
Deploy Script: deployZOS.xml	Integration Stream Name (UCM):
Locked: Yes	
Hidden: No	
SECURITY SETTINGS (OPTIONAL)	
User Access:	
Admin Access:	

Check Project Name in the VCR

Create Reset

During this creation step, IKAN ALM will also automatically create the “Head” Project Stream and the “Base” lifecycle.

Step 3

Step 3: Adapt the lifecycle (if necessary)

By default, a “Base” lifecycle is created for the Project, which can be used for defining the required Build and Deploy Levels (i.e., the logical environments). If that lifecycle is not sufficient for the project, the IKAN ALM Administrator needs to define a new lifecycle.

The following screen shows how, for example, a Build/Compile level is created.

Create BUILD Level

Name: BUILDZOS

Description: Build z/OS for compiling

Type: Build

Locked: Yes

Debug: Yes No

Notification Type: mail

Notification Criteria: Fail

Requester User Group: ALM Project

Life-Cycle: BASE

[Create](#) [Reset](#) [Back](#)

Create Build Level

Once the Build/Compile level is created, it will be displayed on the Life-Cycles Overview screen.

Life-Cycle : BASE

	DESCRIPTION
	BASE Lifecycle

Defined Levels

	NAME	DESCRIPTION	TYPE	LOCKED	OPTIONAL	NOTIFICATION TYPE (CRITERIA)	REQUESTER	PRE-NOTIFY	PRE-APPROVAL
	BUILDZOS	Build z/OS for compiling	Build			mail (Fail)	ALM Project		

[Create Test Level](#) [Create Production Level](#)

Life-Cycles Overview

Step 4

Step 4: Define the environments and the necessary parameters

For each (logical) Level (Build, Test or Production), one or more (physical) environments can be defined. The following screen shows the definition of a Build Environment.

The 'Create Build Environment' dialog box contains the following fields and options:

- Name:** BUILDZOS
- Build Suffix:** (empty)
- Source Location:** D:/ikan/ALM_environments/contbuild/source
- Target Location:** D:/ikan/ALM_environments/contbuild/target
- Build Script:** (empty)
- Downloadable Build:** Yes No
- Debug:** Yes No
- Level:** BUILDZOS (dropdown)
- Machine:** hercikanxp (dropdown)
- Build Tool:** ANT (dropdown)

Buttons: **Create**, **Reset**

Create Build Environment

Once the Environment is created, the IKAN ALM Administrator can define the necessary parameters for this environment. Examples of parameters for the BUILDZOS environment are: the z/OS FTP Server address, the User ID and Password for connecting with the z/OS LPAR.

The 'Search Parameters' dialog box shows the following configuration:

- Build Environment:** BUILDZOS
- Deploy Environment:** (empty)
- Parameter Type:** Build
- Key:** (empty)
- Machine:** (empty)
- Show Machine Parameters:** Yes No
- Mandatory:** Yes No All
- Editable:** Yes No All
- Dynamic:** Yes No All
- Secure:** Yes No All

Buttons: **Search**, **Reset**

The 'Parameters Overview' table shows:

ENVIRONMENT	TYPE	MACHINE	ACTIONS	KEY	VALUE	DES
BUILDZOS	Build	hercikanxp				

Message: No Parameters found

The 'Create Environment Parameter' dialog box shows the following configuration:

- Environment:** BUILDZOS
- Type:** Build
- Secure:** Yes No
- Key:** ftp.tcpip *
- Value:** 192.168.253.168
- Description:** z/OS FTP Server
- Mandatory:** Yes No
- Editable:** Yes No
- Dynamic:** Yes No

Buttons: **Create**, **Reset**, **Cancel**

Environment, Parameters

Step 5

Step 5: Add phases

Once the Levels and Environments have been created, the IKAN ALM Administrator can define the Build or Deploy process by adding Phases. When an Environment is being created, IKAN ALM adds, by default, the IKAN ALM core Phases.

For the z/OS platform, the applicable z/OS Phases have to be added in the correct order for execution. Two z/OS Phases need to be inserted: z/OS Copy Sources before Compilation and z/OS programs Compilation.

Build Environment

Name BUILDZOS	Downloadable Build No
Build Suffix	Debug No
Source Location D:/ikan/ALM_environments/contbuild/source	Level BUILDZOS
Target Location D:/ikan/ALM_environments/contbuild/target	Machine hercikanxp
Build Script	Build Tool ANT

Phase to insert

Phase z/OS Copy Sources before Compilation - 1.0.0

Fail on Error Yes No

Next Phase on Error 5. Cleanup Source - 5.5.0

Insert at position 2

Phases Overview

Position	Phase
1	Transport Source - 5.5.0
2	Transport Deploy Script - 5.5.0
3	Compress Build - 5.5.0
4	Archive Result - 5.5.0
5	Cleanup Source - 5.5.0
6	Cleanup Result - 5.5.0

Available Phases

Phase Name	Phase Version	Execution Type	Author
<input type="radio"/> z/OS Collect Idms components	1.0.0	ANT	IKAN
<input checked="" type="radio"/> z/OS Copy Sources before Compilation	1.0.0	ANT	IKAN
<input type="radio"/> z/OS programs Compilation	1.0.0	ANT	IKAN
<input type="radio"/> Execute Script	5.5.0	CORE	IKAN
<input type="radio"/> Cleanup Result	5.5.0	CORE	IKAN
<input type="radio"/> Cleanup Source	5.5.0	CORE	IKAN
<input type="radio"/> Archive Result	5.5.0	CORE	IKAN

Phase to insert

Once inserted, they need to be put in the right order of execution: first the sources to be compiled have to be transferred to the mainframe and after that the compile process can be started.

Phases Overview						
			PHASE NAME	PHASE VERSION	FAIL ON ERROR	NEXT PHASE ON ERR
	↓	✎	✖ Transport Source	5.5.0	Yes	Cleanup Source
	↓	✎	z/OS Copy Sources before Compilation	1.0.0	Yes	Cleanup Source
	↓	✎	z/OS programs Compilation	1.0.0	Yes	Cleanup Source
	↑	✎	✖ Transport Deploy Script	5.5.0	Yes	Cleanup Source
	↑	✎	✖ Compress Build	5.5.0	Yes	Cleanup Source
	↑	✎	✖ Archive Result	5.5.0	Yes	Cleanup Source
	↑	✎	✖ Cleanup Source	5.5.0	No	Cleanup Result
	↑	✎	✖ Cleanup Result	5.5.0	No	

Phase Overview, right order

Integrating IKAN ALM and Mainframes 31

Step 6

Step 6: Modify the phase parameters

Each Phase comes with Default values, set by the IKAN ALM Administrator at import.

If required, the default values of these parameters can be modified as shown on the Phase Parameters screen below.

Environment Phase

Phase Name z/OS programs Compilation	Next Phase on Success Transport Deploy Script
Phase Version 1.0.0	Next Phase on Error Cleanup Source
Fail on Error Yes	

[Phases Overview](#)

Phase Parameters

			NAME	VALUE	INTEGRATION TYPE	MANDATORY	SECURE
			alm.phase.builder		ANT	✓	
			alm.phase.extractBundle	true	None	✓	
			alm.phase.mainScript	zosCompilation.xml	None	✓	
			dir.zosModels		None		
			dir.zosProperties	D:/ikan/ALM_system/phaseProps	None		
			ftp.active		None		
			ftp.password		None		✓
			propsfile.environment		None		
			propsfile.extensions		None		
			propsfile.languages	\${dir.zosProperties}/languages.properties	None		
			propsfile.parmsFTP		None		

11 items found, displaying all

Phase parameters

Using the same method, the Deploy Environment is completed with the required z/OS deployment Phases such as: 'Promote components and load-modules to z/OS', 'z/OS Delete Sources and associated files', 'z/OS DB2 Binds transfer and activation', 'z/OS Cics Load-modules activation'.

Deploy Environment	
Name ZOSTEST	Debug No
Source Location D:/ikan/ALM_environments/testdeploy/source	Level ZOSTEST
Target Location D:/ikan/ALM_environments/testdeploy/target/ZOSDEMO	Machine hercikanxp
Deploy Script	Deploy Tool ANT
Partial Deploy No	Build Environment BUILDZOS

Phases Overview					
		PHASE NAME	PHASE VERSION	FAIL ON ERROR	NEXT PHASE ON ERROR
		Transport Build Result	5.5.0	Yes	Cleanup Build Result
		Decompress Build Result	5.5.0	Yes	Cleanup Build Result
		Promote components and load-modules to z/OS	1.0.0	Yes	Cleanup Build Result
		z/OS Delete Sources and associated objects	1.0.0	Yes	Cleanup Build Result
		z/OS DB2 Binds transfer and activation	1.0.0	Yes	Cleanup Build Result
		z/OS Cics Load-modules activation	1.0.0	Yes	Cleanup Build Result
		Cleanup Build Result	5.5.0	No	

[Insert Phase](#) [History](#)

Next, the IKAN ALM Administrator changes some default parameters for its target environment:

Environment Phase	
Phase Name Promote components and load-modules to z/OS	Next Phase on Success z/OS Delete Sources and associated objects
Phase Version 1.0.0	Next Phase on Error Cleanup Build Result
Fail on Error Yes	

[Phases Overview](#)

Phase Parameters						
		NAME	VALUE	INTEGRATION TYPE	MANDATORY	SECURE
		alm.phase.builder		ANT	✓	
		alm.phase.extractBundle	true	None	✓	
		alm.phase.mainScript	zosPromotion.xml	None	✓	
		dir.zosModels		None		
		dir.zosProperties		None		
		ftp.active		None		
		ftp.password		None		✓
		propsfile.environment	\${dir.zosProperties}/environment_test_promote.properties	None		
		propsfile.extensions		None		
		propsfile.parmsFTP		None		

10 items found, displaying all



Now that the IKAN ALM Administrator has done his job, the User can start using IKAN ALM for building/compiling, promoting or deploying his programs.

Conclusion



IKAN ALM offers an alternative for pure mainframe-based development by combining an Eclipse-based development environment with a distributed version control repository. On top of that IKAN ALM complements the development process with Application Lifecycle Management and deploy services.

IKAN ALM's major asset is its concept of Phases. JCL can be very complicated. By using IKAN ALM Phases, you can generate and tailor any JCL step.

Thanks to the phase concept and the available models and resources we can also guarantee an easy and successful implementation (as an average, it will only take a few weeks).

The key element is for you to define your ALM process. Once that has been established, the implementation of IKAN ALM is fast and straightforward.

If you are already using a mainframe solution like CA-Endevor or Serena ChangeMan and you would decide to migrate to IKAN ALM, you will of course need to migrate your CA-Endevor or Serena ChangeMan legacy to IKAN ALM. To do so, we have a standard migration procedure.



In a nutshell: by implementing IKAN ALM, you can continue exploiting the full strengths of your mainframe and seamlessly combine them with new innovative tooling. This will help you cutting down the costs of maintaining different systems, and above all ease the work of your developers as IKAN ALM will take care of the different steps in the lifecycle of your application including its deploy on the mainframe

For More Information

To know more, visit <http://www.ikanalm.com>

Contact IKAN Development: info@ikanalm.com

Related Document

Modern Mainframe Development and ALM

The following appendices explain the terminology used by the different ALM mainframe software providers.

Appendix I: IKAN ALM Terminology

The following table explains the terms used by IKAN ALM and provides a brief comment for each of them. This will help users of IKAN ALM to have a better understanding of the terminology used.

IKAN ALM	Remarks
VCR	A Version Control Repository contains the components to manage. Examples of VCRs are: CVS, Subversion, IBM ClearCase, Serena PVCS, Microsoft VisualSourceSafe.
Project	A tailored Lifecycle process including development, testing, quality assurance and production can be easily defined, implemented and enforced, offering a comprehensive framework across all major platforms including Windows, UNIX, Linux and IBM mainframe systems. IKAN ALM also supports a stream-based project model allowing project managers to easily add Lifecycles to each version of a project, which makes it easy to differentiate between maintenance, “urgency fix” or release build and deploy processes.
Lifecycle	Defines the Lifecycle(s) from Development to Production Levels for Streams.
Project Stream	Each IKAN ALM Project contains exactly one HEAD Project Stream and may contain one or more Branches. A Project Stream is a working entity within IKAN ALM
Level	Defines every step of the Lifecycle from Development to Production, supporting physical Environments.
Environment	IKAN ALM uses the (logical) Level concept in which (Build/Deploy) environments can be defined. Every environment represents a Machine (Server/OS) on the network where the Source and Target Locations are defined for executing Phases. This is a unique architectural IKAN ALM feature, representing the true multi-platform aspect of IKAN ALM.
Package	Instead of using a Release Project, IKAN ALM may work with Packages in the Project. The required files must be added manually from the VCR to the Package IKAN ALM does not contain the Sources. It only knows the link to the files in the VCR project.
Level Request	A Level Request in IKAN ALM starts a Build, Deploy and Rollback in the Environment(s).
Build Request	The Build Level Request type in IKAN ALM will usually take care of a compile procedure for components.
Phase	users can extensively customize the workflow of their projects, by using highly reusable building blocks, called Phases. By using the import/export features, Phases can not only be shared between different Projects, but also between different IKAN ALM installations.
Script (Ant)	Runs the process (e.g., for build, compilation, deployment, copy, etc.) using the Source and Target locations on the Agent machine. A script can use property files, models and other scripts generally defined in a Phase.
Build#	IKAN ALM generates a unique build number that can be used in several processes to identify the output from the (build/compile) procedures. IKAN ALM is also able to put this information on members in a Partitioned Dataset on z/OS.

Approval	IKAN ALM allows setting up a hierarchy in the Approval Groups. For example, Group2 may only approve if Group1 has approved first. Groups are based on the Enterprise Security System users.
Rollback	In IKAN ALM, an automatic rollback can be executed for every kind of output, which will allow the customer to completely automate a rollback operation. Typically, it runs a previous version of your choice.
Machine	A machine runs an IKAN ALM Agent which will take care of building/deploying the software components. Linux/UNIX (flavors) and Windows platforms. The Agent (LUW) machine can update more than one LPAR via FTP connections. Z/OS Phases might be reused with models and PDS definitions using several FTP connections.
Release Number/ Incident Number	IKAN ALM has an ITS-plugin that allows you to easily link existing issue or defect tracking systems. Issues are accumulated along the Lifecycle and updated automatically.
Archive	IKAN ALM compresses and saves all Build results in Archives and keeps them in a dedicated location. Archives are identified with the Build Tag.
Report	IKAN ALM has a web interface to view per project and level request what happened. On top of that, an ALM-Reports tool allows creating more Reports about Global and Project Administration and Package activities.
Phase adds	Remarks
Extension/ Object-type	The extension/objtype determines the processing needed for a certain file type. This is defined by a property file and scripts. Object-types are used for z/OS activities.
Obsolete File	IKAN ALM has no process for scratching individual files. This action is resolved with an Environment Phase which scratches the source component using the “.to_be_deleted” suffix in the VCR. Associated z/OS components are deleted in their PDS.

Appendix II: CA-ENDEVOR Terminology

The following table maps the terms used by IKAN ALM and CA-Endevor and provides a brief comment for each of them. This will help the respective users of IKAN ALM or CA-Endevor to have a better understanding of the terminology used.

IKAN ALM	CA Endevor	Remarks
VCR	Database/Delta	CA-Endevor assumes VCR versions with the Image and Delta file(s).
Project	System and/or Sub-system	Within IKAN ALM, the defined project needs attributes to tell IKAN ALM to which CA-Endevor System/Sub-system the Software Items should be added in Environment parameters.
Lifecycle	Map	Defines the Lifecycle from Development to Production.
Stream	Not available	CA-Endevor works with a unique Project version.
Package	Package	CA-Endevor groups components in Batch packages.
Level	Stage	Defines every step of the Lifecycle from Development to Production.

Environment	(Stage)	This is a unique architectural IKAN ALM feature, not known in CA-Endevor, representing the true multi-platform aspect of IKAN ALM.
Level Request	(Move) Action	CA-Endevor distinguishes more actions, but they are not directly applicable to IKAN ALM. For example, to delete a component from the Production environment, the components should be deleted from the VCR; the project should be built and deployed, and tested in all the Levels between Development and Production, ensuring that this deletion does not jeopardize the Production. This delete task may be activated with a SVN property on the component (don't delete). Next, IKAN ALM may assume the deletion during the deployment by using the SVN property in a script.
Build Request	(Add) Action	The Build Level Request in IKAN ALM will usually take care of populating CA-Endevor with the Software Components (ADD action).
Phase	Processor group	The processor group in CA-Endevor determines the ultimate process to run within a certain type. For example, the Processor Type COBOL might have processor groups for COBOL, DB2, CICS, BATCH, IMS etc.
Script (Ant)	Processor	Runs the process (e.g., for build or compilation).
Idrdata/Build#	Footprint	Build# or Build number: is an incremental number given after each software build. IDR DATA: Identification record data field. Identification records have a fixed format and fixed content, both defined by the program management binder. Is used by IBM Endevor footprints contain the following information: site ID, environment, system, subsystem, element, type, stage, version/level, and generate date/time.
Approval	Approval	CA-Endevor allows defining several Approval Groups which are in the same hierarchy. Every group may approve on any moment.
Rollback	Backout	CA-Endevor allows reversing the result from a promotion/delivery if it is a member(s) in a Partitioned Dataset (PDS). In the case of DB2 a (manual) rebind should be executed.
Machine	Ship	CA-Endevor only supports other z/OS Logical Partitions (LPARS).
Release Number/ Incident Number	CCID	The release/incident number within IKAN ALM may be passed to CA-Endevor as the CCID (Change Control Identifiers) most often correspond to mechanisms such as work order requests or request-for-service numbers.
Archive	Not available	CA-Endevor keeps these components in Stage Level with CCID's.
Report	Report	CA-Endevor allows using Batch reports.
Extension/ Object-type	Type	The extension/objtype determines the processing needed for a certain type.

Appendix III: Serena ChangeMan ZMF terminology

The following table maps the terms used by IKAN ALM and Serena ChangeMan ZMF and provides a brief comment for each of them. This will help the respective users of IKAN ALM or ChangeMan to have a better understanding of the terminology used.

IKAN ALM	ChangeMan ZMF	Remarks
VCR	Baseline/Delta/Package	ChangeMan assumes VCR functionalities as Check-Out, Commit (Baseline Ripple), Check-In (Freeze) from the Package Lifecycle.
Project	Application	ChangeMan has the same concepts as IKAN ALM but only for IBM mainframe systems. Also the stream-based project is not available.
Lifecycle	Stage/ Promotion Levels	Defines the Lifecycle from Development to Production.
Stream	Not available	ChangeMan works with a unique Project version.
Package	Package	ChangeMan uses the Package for the Development process up to the Stage action. IKAN ALM leaves development actions to the customer IDE and the VCR. Both IKAN ALM with Level Requests and ChangeMan with Staging, manage Build (Compile) requests, as well as Deployments with the Approval supervision for the Package. ChangeMan, however, needs to update the Baseline & Stacked Reverse Delta supports in double with the Production and the Package.
Level	Promotion Level	Defines every Level of the Lifecycle from Development to Production.
Environment	Site (Local or Remote)	The Local or Remote Site concept in ChangeMan is covered by the IKAN ALM environment concept. An IKAN ALM level (a logical step) can have one or more environments.
Phase	Procedures or skeletons	The skeletons in ChangeMan determine the process to run within a certain type. For example, the Procedure CMNCOB2 might have process skeletons for COBOL, DB2, CICS, IMS, etc.. Depending on the Source options.
Level Request	Promotion/ Demotion	ChangeMan knows more actions, but they are not directly applicable to IKAN ALM. For example, to delete a component from the Production environment, the components should be renamed into the VCR with the special “.to_be_deleted” suffix. Next, the project should be built and deployed, and tested in all the Levels between Development and Production, ensuring that this deletion does not jeopardize the Production. This delete task will be activated with the suffix of the component (don’t delete). Next IKAN ALM may assume the deletion during the deployment by using the suffix in the dedicated Phase.
Build Request	(ST) Action	ChangeMan takes care of the compile procedure the same way as IKAN ALM.
Script (Ant)	Skeleton Procedure	Runs the process (e.g., for build or compilation or deploy).
Build#	Package Number	ChangeMan uses the Package number for versioning files.
Approval	Approval	ChangeMan allows defining several Approval Groups which are hierarchical. Every group may approve one after the other.

Rollback	Demotion	ChangeMan allows reversing the result from a promotion/delivery if it is a member(s) in a Partitioned Dataset (PDS). In the case of DB2 a (automatic) rebind will be executed.
Machine	Site	ChangeMan only supports other z/OS Logical Partitions (LPARS). The ChangeMan site is the Local or a Remote LPAR.
Release Number/ Incident Number	Not available	ChangeMan does not use Incident numbers. In the Package description panel, a reason may be entered for all included components.
Archive	Package	ChangeMan contains components in the Package which is frozen before the deployment. It designs the version to deploy.
Report	Report	ChangeMan allows using Batch reports.
Obsolete File	Scratch/ Rename	ChangeMan assumes the Scratch and the Rename functionalities during the Promote. Will be supported through a custom phase.
Object-type	Library type	The objtype determines the processing needed for a certain type. IKAN ALM can use the same codes.
IKAN Impact Analysis Tool	Impact-Analysis	ChangeMan Impact Analysis covers Source, Copy, JCL, Proc and DSN names relationships. The Impact Analysis solution from IKAN ALM, permits you to create an Impact Analysis table, based on the information available in the Version Control Repository. IKANALM reports are available to query that Impact Analysis table and as such you get the same and more results as with the Change Man Impact Analysis solution.
Not available	Merge & Reconcile	IKAN ALM does not need to support this because it is a task of the VCR.
As Archive	Freeze	In relation to this ChangeMan concept, IKAN ALM creates an Archive at the end of every Build containing all components to deploy. It is this Archive that used for the next.
Not available	Baseline	It is a ChangeMan concept that duplicates (or not) the Production Level used for future package developments considering it is the version 0 as Reference in Production. IKAN ALM doesn't assume this concept because it is the VCR task to define the versions of components. IKAN ALM creates or presents the Tag for a Build version.

Appendix IV: Available z/OS IKAN ALM Phases

The following table maps the Phases used by IKAN ALM for compiling and deploying components to Mainframe Environments. Note that for IDMS, a Phase collects the dictionary components and the next phase installs them into another one.

Phase	Action	Description
z/OS copy Source to Target	Build/ Deploy	A dedicated Phase for copying the z/OS Components (Sources or Objects) to the IKAN ALM Target Environment. This Phase only transfers selected component types.
z/OS copy Sources before Compilation	Build	This Phase transfers via FTP Copybooks, Linkedit Control Cards (LCT cards) Assembler, COBOL PL/1 Programs and BMS/SDF2 Maps to PDS(s) in the Mainframe Environment.

z/OS Programs Compilation	Build	For each Map (firstly) and each Program (secondly), the Phase generates a compile JCL depending on the Source contents and language using included JCL models. Next, each JCL is executed by JES under FTP and the resulting Job is analyzed to know its status. Next, the generated compile Listing, Load-module and DB2 DBRM, Datacom Plan are transferred to the IKAN ALM Target Environment. Optionally, DB2 Binds may be generated from models. Note that for CA-Endevor the Repository will be updated for compiling with it.
z/OS Promotion of components and load-modules	Deploy	z/OS components in the IKAN ALM archive are transferred to their PDS(s) of the Mainframe Environment. Exception: the Load-modules which are transferred to flat files before a generated JCL using included JCL models is executed by JES under FTP for receiving them in their PDS(s).
z/OS Delete Sources and associated objects	Deploy	All Sources identified by the “to_be_deleted” suffix are deleted in PDS(s) of the Mainframe Environment by FTP. Also, the associated Listings, Load-modules, DBRMs, Plans and DB2 Binds are deleted in their PDS(s). No action in DB2 and Datacom Databases.
z/OS DB2 Binds transfers and activation	Deploy	If DB2 is used, Bind files are copied to their PDS(s) and a JCL is generated using included JCL models and executed by JES under FTP for running these Binds on the DB2 Database.
z/OS CICS Load-modules activation	Deploy	If there are CICS Maps or Programs, a JCL is generated using included JCL models, and executed by JES under FTP for running the PHASEIN commands on a CICS.
z/OS Update Datacom components Promotion	Deploy	If there are Plans, the Phase generates a JCL using included JCL models and executed by JES under FTP for importing Plans on the Datacom Database.
z/OS Update Endevor components Promotion	Deploy	If the CA-Endevor Repository is active on the Mainframe, the Phase generates a JCL using included JCL models and executed by JES under FTP for moving components from the Stage ID to the corresponding Level.
z/OS SQL DB2 updates Execution	Deploy	If DB2 is used, DDL and SQL statements may be applied with variable substitutions as owner, qualifier. After the transfer of DDL and SQL commands concatenated into 2 members, 2 JCLs are generated using included JCL models and executed by JES under FTP for running DDL and, next, SQL on the DB2 Database.
z/OS Update Debugger	Deploy	For instance, for the Xpediter tool, this phase copies Xpediter components from a FILEIO file to another FILEIO using the components list in the IKAN ALM target environment.
z/OS Copy Pds Members	Deploy	This phase transfers components from PDS(s) of a z/OS environment to PDS(s) of another z/OS environment using the components list in the IKAN ALM target environment.
z/OS Update QMF	Deploy	This phase imports QMF components to a QMF DB2 sub-system using the components list in the IKAN ALM target environment.
z/OS Collect IDMS components	Build	For the first build, this Phase generates a JCL using included JCL models that is executed in the Mainframe Environment by JES under FTP. This one collects IDMS components and info about date and parent relations in the IDD of development. For the rebuild before deployment, the Phase controls the correlation with the target IDD with execution of another generated JCL.

z/OS IDMS components Promotion	Deploy	This phase transfers components to temporary files in the Mainframe Environment and she generates a JCL using included JCL models that is executed by JES under FTP for updating the target IDD.
... (on demand)		

Appendix V: Migration to IKAN ALM

Before you can work with IKAN ALM, components must be installed in a VCR. This is a big difference with CMN where the full VCR is included in Packages, Baseline and SRDeltas PDSs, or with Endeavor where components are in workspace PDSs.

IKAN has developed an Ant solution for migrating components from the CMN versioning system to VCR projects (Subversion or Clearcase). The Tool supports the collect of versions in the SRDeltas, Baseline and Packages for all types of components based on the CMN project concept. The results are the same versioning levels in the VCR Projects and same Package definitions that you had in CMN.

The solution supports migrations from classic PDS.

IKAN also has developed an Ant solution for updating VCR Projects and IKAN ALM Packages from other tools (z/OS tools or Windows/Unix tools) to automatically version and deploy some components using the package process.

Appendix VI: Sample of z/OS compilation JCL

The following JCL is fully generated by the z/OS Compilation Phase used by IKAN ALM for compiling a component into the Mainframe Environment.

```
//ADCDMSTC JOB (5145,00000,2233,T),'IKAN',
//          MSGLEVEL=(1,1),MSGCLASS=X,
//          CLASS=A,REGION=8M
//*
//*XEQ ROUTEID=ADCD
//*****
/**  COPYING THE PROGRAM IN SOURCE WORK FILE          **
//*****
//      SET SRCOMPIL=SOURCE
//COPYSRC EXEC PGM=IEBGENER
//SYSTSIN DD DUMMY
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUT1  DD DISP=(SHR),
//          DSN=IKANALM.DEMOS.TEST.SRCBATCH(DEMO21)
//SYSUT2  DD DISP=(,PASS),DSN=%%&&SRCOMPIL,
//          UNIT=SYSDA,SPACE=(CYL,(10,10)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSIN   DD DUMMY
//*****
/**  COMPILER COBOL2, store object in objlib if compile=ok
```

```

/**      compile listing is stored in IKANALM.DEMOS.TEST.LSTALIB
/*****
//          SET PARMCOB='LIST,LIB,NOSEQ,NOCMR2,MAP'
//          SET PARMCOB0='DATA(31)'
/*****
/**      COMPILER THE ELEMENT      **
/*****
//COBOL EXEC PGM=IGYCRCTL,COND=(4,LT),
//  PARM='&PARMCOB0,&PARMCOB'
//STEPLIB DD DISP=SHR,DSN=SYS1.COB2COMP
//SYSIN DD DISP=(OLD,PASS),DSN=*&&SRCOMPIL
//SYSLIN DD DISP=(,PASS),DSN=*&&OBJECT,
//          UNIT=SYSDA,SPACE=(CYL,(2,2)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSUT7 DD UNIT=SYSDA,SPACE=(CYL,(5,3))
//SYSPRINT DD DISP=(,PASS),DSN=*&&COMPLIST,
//          UNIT=SYSDA,SPACE=(TRK,(10,10),RLSE)
/**      DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
/**
//SYSLIB DD DISP=SHR,DSN=IKANALM.DEMOS.TEST.COPYLIB
//          DD DISP=SHR,DSN=IKANALM.DEMOS.INTG.COPYLIB
//          DD DISP=SHR,DSN=IKANALM.DEMOS.QUAL.COPYLIB
//          DD DISP=SHR,DSN=IKANALM.DEMOS.PROD.COPYLIB
/**
/**
/**
/**
/**
/**
/*****
/**      COPYING THE LCT MEMBER IN A WORK FILE IF EXIST      **
/*****
//ALLOCLCT EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//LCTFILE DD DISP=(NEW,PASS,DELETE),DSN=*&&LCTFILE,
//          UNIT=SYSDA,SPACE=(TRK,(1,1,1)),
//          DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0)
//CREATLCT EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD *
/*
//SYSUT2 DD DISP=(MOD,PASS),DSN=*&&LCTFILE(DEMO21)
//SYSIN DD DUMMY
//COPYLCT EXEC PGM=IEBCOPY

```

```

//SYSPRINT DD SYSOUT=*
//INDD00 DD DISP=SHR,DSN=IKANALM.DEMOS.TEST.LCTLIB
//INDD01 DD DISP=SHR,DSN=IKANALM.DEMOS.INTG.LCTLIB
//INDD02 DD DISP=SHR,DSN=IKANALM.DEMOS.QUAL.LCTLIB
//INDD03 DD DISP=SHR,DSN=IKANALM.DEMOS.PROD.LCTLIB
//*
//OUTDD1 DD DISP=(MOD,PASS),DSN=&&LCTFILE
//SYSIN DD *
COPY OUTDD=OUTDD1
INDD=INDD00,INDD01,INDD02,INDD03
SELECT MEMBER=DEMO21
/*
//*****
//** LINKEDIT PROGRAM **
//*****
// SET PARMLNK='LIST,MAP,XREF,NCAL'
// SET LINKOPT='RENT,AMODE(31),RMODE(ANY),'
//LKEDT EXEC PGM=HEWL,COND=(4,LT),
// PARM='&PARMLNK,&LINKOPT'
//SYSLMOD DD DISP=SHR,DSN=IKANALM.DEMOS.TEST.LOADLIB(DEMO211)
//SYSDEFSD DD DUMMY
//SYSPRINT DD DISP=(,PASS),DSN=&&LINKLIST,
// UNIT=VIO,SPACE=(TRK,(10,10)),
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=0)
//* DD DISP=SHR,DSN=IKANALM.DEMOS.TEST.LOADLIB(DEMO211)
//SYSLIB DD DISP=SHR,DSN=IKANALM.DEMOS.TEST.LOADLIB
// DD DISP=SHR,DSN=IKANALM.DEMOS.INTG.LOADLIB
// DD DISP=SHR,DSN=IKANALM.DEMOS.QUAL.LOADLIB
// DD DISP=SHR,DSN=IKANALM.DEMOS.PROD.LOADLIB
//*
// DD DISP=SHR,DSN=DFH320.CICS.SDFHLOAD
// DD DISP=SHR,DSN=DSN810.SDSNLOAD
// DD DISP=SHR,DSN=CEE.SCEELKED
//* DD DISP=SHR,DSN=METASUIT.GEN813.LOADLIB
//* DD DISP=SHR,DSN=SYS1.COB2LIB
// DD DISP=SHR,DSN=SYS1.LINKLIB
//SYSLIN DD *
/*
// DD DSN=&&OBJECT,DISP=(OLD,DELETE)
// DD DSN=&&LCTFILE(DEMO21),DISP=(OLD,DELETE)
// DD *
IDENTIFY DEMO211('DEMO21/ADCDMST/000003')
NAME DEMO211(R)
/*
//*****
//** TRANSMIT PROGRAM **
//*****
//CLEARSEQ EXEC PGM=IEFBR14
//DD01 DD DISP=(MOD,DELETE,DELETE),
// DSN=IKANALM.DEMOS.TEST.DEMO211,

```

```

//          UNIT=SYSDA,SPACE=(TRK,(1)),
//          LRECL=80,BLKSIZE=3120,RECFM=FB
//*
//XMITLOAD EXEC PGM=IKJEFT01,COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
XMIT (ADCD.*) -
DSNAME('IKANALM.DEMOS.TEST.LOADLIB') MEM(DEMO211)-
OUTDSNAME('IKANALM.DEMOS.TEST.DEMO211') NOLOG NONOTIFY
/*
//PRTCMPA IF (COBOL.RUN EQ TRUE) THEN
//*****
//** PRINT THE COMPILE LISTING **
//*****
//PRNTCOMP EXEC PGM=IEBGENER
//SYSTSIN DD DUMMY
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=(OLD,PASS),DSN=&&COMPLIST
//SYSUT2 DD SYSOUT=*
//SYSIN DD DUMMY
//PRTCMPZ ENDIF
/*
//PRTLKKA IF (LKEDT.RUN EQ TRUE) THEN
//*****
//** PRINT THE LINKEDIT LISTING **
//*****
//PRNTLINK EXEC PGM=IEBGENER
//SYSTSIN DD DUMMY
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=(OLD,PASS),DSN=&&LINKLIST
//SYSUT2 DD SYSOUT=*
//SYSIN DD DUMMY
//*****
//** FORMAT THE LINKEDIT LISTING **
//*****
//IFSFTLKD IF (NOT ABEND) THEN
//FRMTLKD EXEC PGM=SORT
//SORTSNAP DD SYSOUT=*
//SORTWK01 DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SORTWK02 DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SORTWK03 DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SORTWK04 DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//SORTIN DD DISP=(OLD,DELETE),DSN=&&LINKLIST
//SORTOUT DD DISP=(NEW,PASS,DELETE),DSN=&&LISTLKD,
//          UNIT=SYSDA,SPACE=(CYL,(5,5)),
//          DCB=(DSORG=PS,RECFM=FBA,LRECL=133,BLKSIZE=0)
//SYSIN DD *

```

```

SORT FIELDS=COPY
OUTREC FIELDS=(1,121,12X)
/*
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//IFEFTLKD ENDIF
//*****
//** COPY THE LISTINGS **
//*****
//IFSLST1 IF (NOT ABEND) THEN
//LIST EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD DISP=SHR,
// DSN=IKANALM.DEMOS.TEST.LSTALIB(DEMO21)
//SYSIN DD DUMMY
//SYSUT1 DD DISP=(NEW,DELETE,DELETE),DSN=&&NULLSEQ,
// UNIT=SYSDA,SPACE=(TRK,(1,1)),
// DCB=(DSORG=PS,RECFM=FBA,LRECL=133,BLKSIZE=0)
//* DD DISP=(OLD,DELETE),DSN=&&PCMLIST
// DD DISP=(OLD,DELETE),DSN=&&COMPLIST
// DD DISP=(OLD,DELETE),DSN=&&LISTLKD
//IFELST1 ENDIF
/*
/*
//IFSFAIL IF (RC GT 4 OR ABEND) THEN
//FAILURE EXEC PGM=IEBGENER,MAXRC=0
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD *
JOB 1626 FAILED
/*
//SYSUT2 DD SYSOUT=*
//SYSIN DD DUMMY
//IFEFAIL ENDIF
//PRTLKDDZ ENDIF

```

IKAN Development N.V.
Kardinaal Mercierplein 2
2800 Mechelen
Tel. +32 15 797306
info@ikan.be
www.ikan.be

© Copyright 2013 IKAN Development N.V.

The IKAN Development and IKAN ALM logos and names and all other IKAN product or service names are trademarks of IKAN Development N.V. All other trademarks are property of their respective owners. No part of this document may be reproduced or transmitted in any form or by any means, electronically or mechanically, for any purpose, without the express written permission of IKAN Development N.V.

IKAN