



Life Cycle Management for Oracle Data Integrator 11 & 12



Increase productivity

Stop wasting your time doing things manually by automating every step in your project's Life Cycle



At lower cost

Get a 30% return on investment guaranteed and save 15% on development costs



For all stakeholders

Developers, Project Managers and Test Engineers are all part of the development process



RedBridge

Table of contents

Life Cycle Management Components	5
ODI Studio.....	5
VCR4ODI	5
Subversion.....	5
IKAN ALM	5
Workflow Organization.....	6
VCR4ODI.....	6
The RESTORE operation	7
The VERSION (commit) operation.....	8
IKAN ALM	9
IKAN ALM setup	9
Step 1: Defining the Versioning Strategy	9
Step 2: Specifying the Life Cycle of Your Product.....	10
Step 3: Setting up the Build and Deploy Environments.....	11
Step 4: Using Phases and Phase Parameters	11
Step 5: Specifying Environment Parameters.....	13
IKAN ALM Level Requests	13
Conclusion.....	15
For more information	15

Management summary

Oracle Data Integrator (ODI) is part of the Oracle Data Integration Suite. It is used for high-performance data movement and transformation, and uses an Extract, Load and Transform (ELT) architecture.

This architecture is organized around a modular repository, which is accessed in client-server mode by components such as ODI Studio and execution Agents.

One of the key challenges for ODI customers is to handle the end-to-end Life Cycle for Oracle Data Integrator.

The current recommended process is complex, labor-intensive and requires manual intervention. As a consequence, it is expensive and it adds a risk of human mistakes because of its complexity.

RedBridge Software provides an automated Life Cycle Management solution, combining ODI, Subversion and IKAN ALM.







Our LCM solution deals with the above mentioned issues and contributes to the business's bottom-line by deploying high-quality applications more frequently and without errors.

The solution takes into account the tools currently being used, the chosen development and deploy processes and the different stakeholders.

It is a common misconception that only developers would profit from Life Cycle Management.

As shown in the table on the next page, all stakeholders will benefit from the implementation of Life Cycle Management. It will also make your organization ready for the implementation of DevOps.



	Development	IT Management	Production	Management/Audit
 Versioning control	Keep track of all code changes	Traceable development status Safe storage of all legacy data	Only use accurate versions Easily revert to a prior version in case of a failure	Cost efficient development Full insight in the development status
 Continuous integration	Get early feedback on committed code Prevent errors from going into test or production	Feedback on code quality Find weak spots Easy fix mistakes	Only high-quality production code. Prevent costly re-runs.	Cost saving by higher efficiency Repeatable and controllable process Faster and shorter release cycle
 Automated build process	No loss of valuable development time by trying to build manually Separation of duties	Unlimited builds Quicker results Rapid feedback	All builds are done correctly according to upfront agreed procedures.	Super cost saver
 Approval management	Improve communication between all members of the project team	Control the evolution in the different stages of the life cycle Build in audit moments	Deployment to the production servers by authorized members only	Full traceability of development and production status Full accountability for made decisions.
 Automated deploy process	Guarantee that production will receive the code as intended	Speed up the process and help reduce errors by automating manual activities	No manual intervention reduces risk	Increase the possible release cycle frequency and improve productivity.
 Rollback process	Defects can be fixed quicker	Always possible to revert to the latest good release	Eliminate the risk of service outage	Quickly resolve errors in production, and, consequently, prevent costly production outfall

In this white paper, we will explain how RedBridge Software's solution ensures end-to-end Life Cycle Management for Oracle Data Integrator.

Life Cycle Management Components

A true Life Cycle Management solution serves the process following the end-to-end principle and takes into account all stakeholders, from development to operations. RedBridge Software's LCM solution uses the following components:

ODI Studio ①

The heart of the solution is the ODI software. Our solution does not impact the current way of developing. Developers will continue to use ODI Studio for their development.

When the development is finished, VCR4ODI will be used to commit the work done to the Version Control Repository.

VCR4ODI ②

To enable the interaction between the ODI Repositories and Subversion, we add an extra component, i.e., the VCR4ODI connector.

That component will be used to commit ODI objects to the VCR and, vice versa, to restore objects from the VCR into ODI.

Subversion ③

Subversion will be used as Version Control Repository.

IKAN ALM 4

Next, we add IKAN's Life Cycle Management Solution, which will give you the possibility to define your Life Cycles.

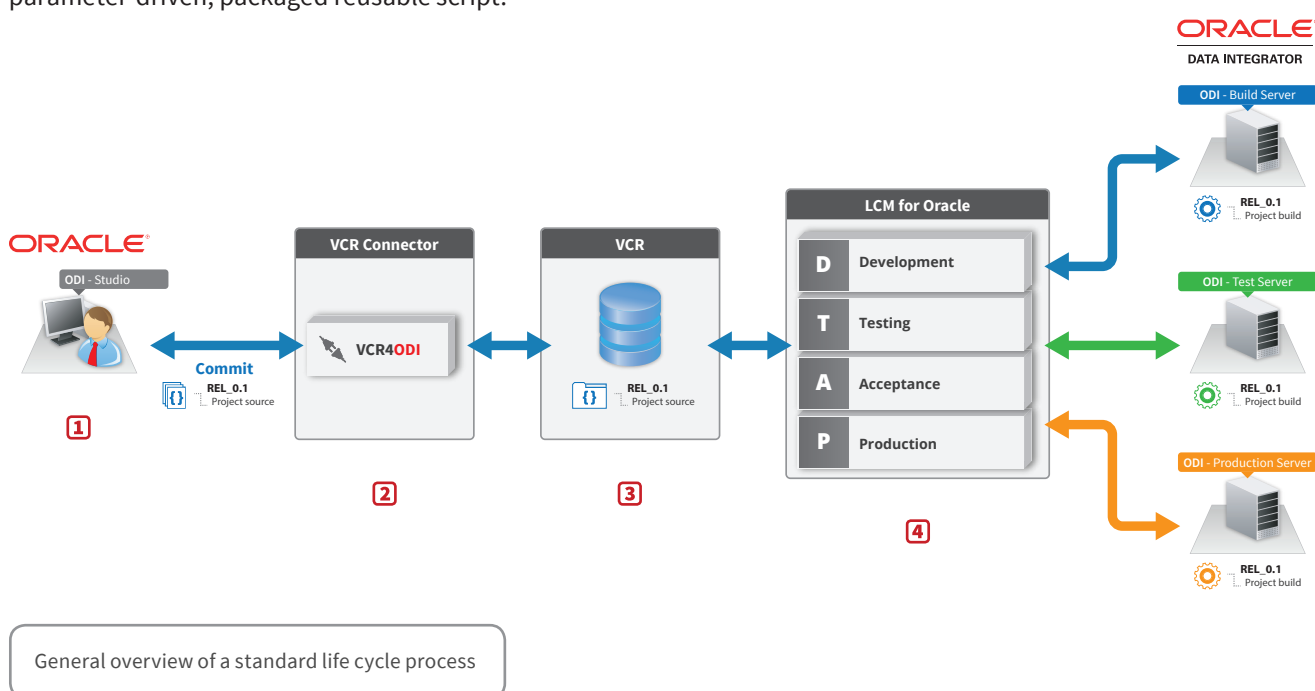
An IKAN ALM life cycle links to a version control repository Trunk (head) for continuous development and to a Branch for the releases. For each Life Cycle, you can define one or more logical Build, Test and Production levels. Within those logical levels, you can define one or more physical environments.

To do the actual Build and Deploy work, IKAN ALM uses the Phases concept. An IKAN ALM phase is a pre-defined, parameter-driven, packaged reusable script.

Specifically for ODI, we have, for example, phases that allow you to create an ODI repository and to deploy ODI files to an ODI repository.

You can use those pre-defined phases, but you can also customize existing phases to fit your needs, e.g., for the Deploy you could add a phase for database changes.

On top of that, IKAN ALM allows you to integrate with other Life Cycle Management components such as requirements and issue tracking systems.



Workflow Organization

When developing software, developers usually work in a team. This means that you can have parallel development and that you need to reconcile all development efforts at a certain point in time. You also need to manage your releases. That is where versioning, or a Version Control Repository, comes into play.

It is important to understand that there are many ways to organize the development process and the usage of Subversion.

In this white paper we will use a generally accepted way of working: for the ongoing development we will use a mainline (called trunk in Subversion) and for each release we will create a separate branch. Possible emergency or hot fixes will be done on the impacted release or branch and will be merged by development to the Subversion mainline or trunk. Simply said, the following steps need to be followed:

- Develop your ELT-programs using ODI Studio
- Use the VCR4ODI connector to check out/commit your ELT-programs to Subversion
- Use IKAN ALM for the, possibly approval-based, Build process, and to execute the Deploy to Test and Production environments

How the development is done in ODI should be clear to you. In the next sections, we will explain in more detail how VCR4ODI and IKAN ALM work.

VCR4ODI

VCR4ODI is a component that connects the ODI repository with Subversion. On the one hand it provides a view on your ODI Repository and on the other hand a view on the Subversion repository.

The VCR4ODI connector allows to

- work at the individual object level so that you do not need to commit or restore a whole project,
- lock objects to prevent others from working on the same objects you are working on,
- restore objects from a previous version.

On top of that, VCR4ODI will calculate all the dependencies between objects, and will automatically create a new version for Projects, Models, Scenarios and the global objects that are used.

When you start up VCR4ODI, you first need to provide the correct properties to connect to the required ODI Repository and to the Subversion trunk or branch.

Providing the properties

VCR4ODI Properties

ODI Connection Parameters

Supervisor User: SUPERVISOR

Supervisor Password:

Master Repository User: ALM_MASTER

Master Password:

Work Repository User: ALM_DEV_WORK

Repository Driver: oracle.jdbc.OracleDriver

Master Repository URL: jdbc:oracle:thin:@127.0.0.1:1521:orcl1107

Subversion Connection Parameters

URL: https://ikandemo004v.ikan.local/svn/ikanalm_test/trunk

User: ikanalm

Password:

Root Temp Dir: E:\ODI_DevMasterWC_ikanalm/

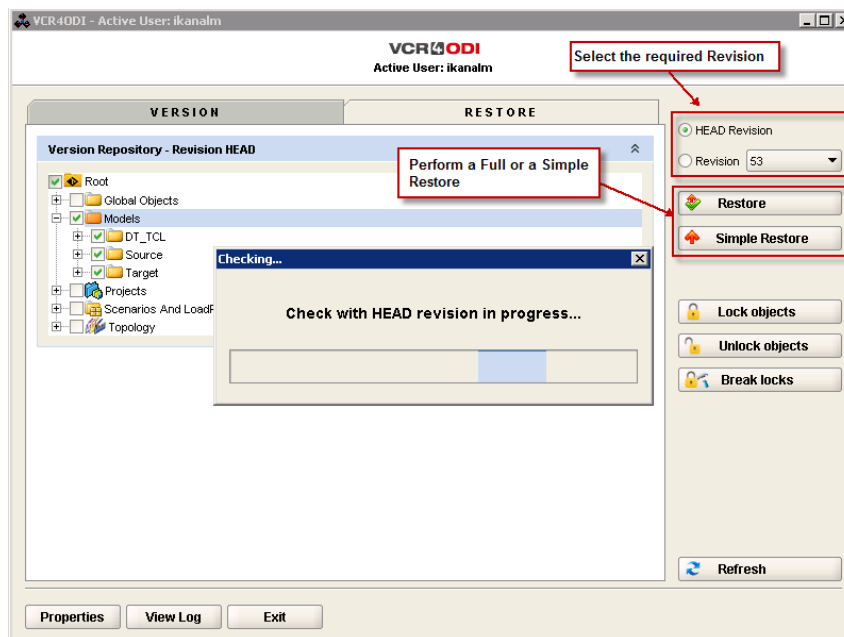
TEST Close Save

Once the connection is established, there are two possible actions:

- **VERSION** (commit) modified ODI objects from the selected ODI Repository into the Subversion Head (trunk) or Revision.
- **RESTORE** objects from Subversion into the selected ODI repository.

The RESTORE operation

You can restore objects from the head or from a specific revision. Additionally, you can choose to execute a (full) restore or a simple restore.

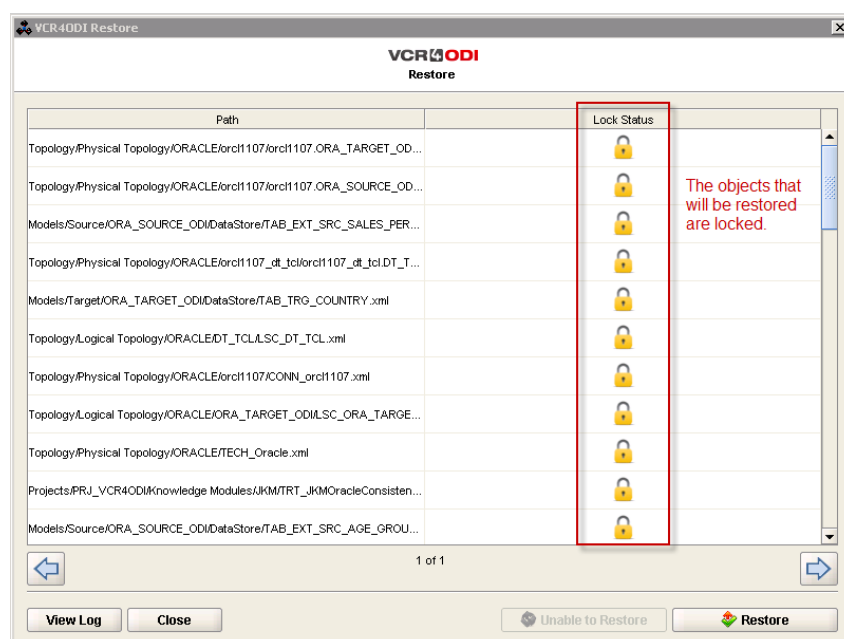


A (full) Restore will restore the selected object(s) and all logically related objects.

A Simple Restore will only restore the selected object. This option is used if you want to make a quick and small change for which you do not need all related objects.

When the restore is completed, the selected objects are locked to prevent concurrent updates.

Full or Simple restore



The selected objects are locked to prevent concurrent updates.

The VERSION (commit) operation

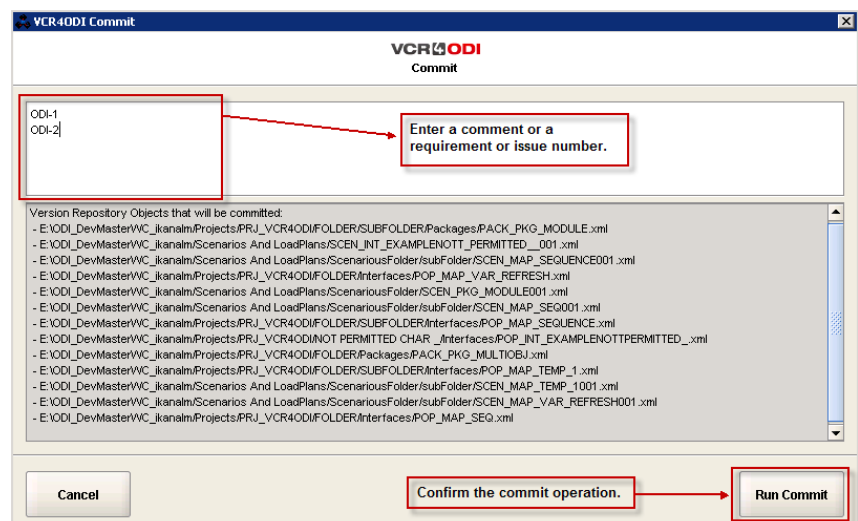
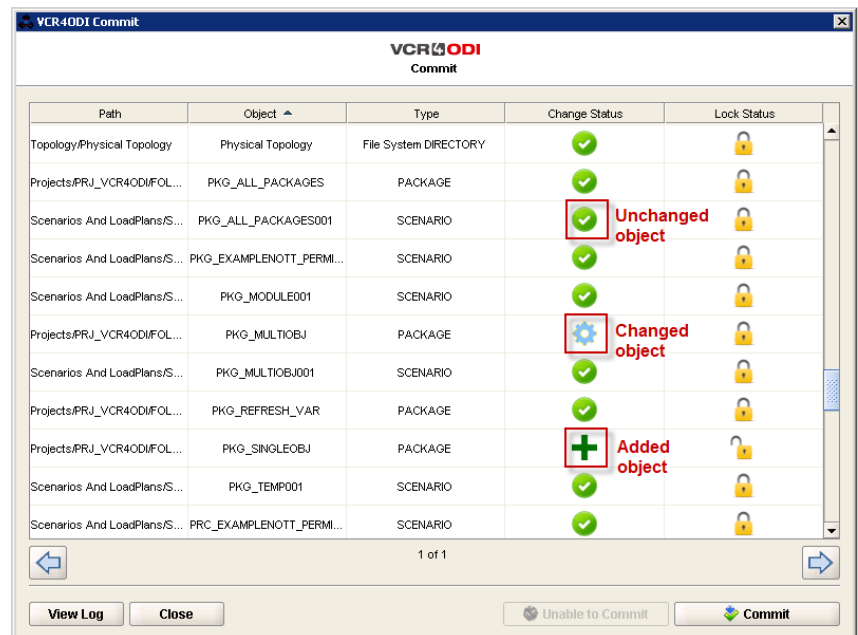
On the VERSION tab page, you can select the objects you want to commit. The VCR4ODI connector applies the following intelligence:

- Objects that are logically connected to a selected object will also be taken into account.
- Only content changes will be considered. For example: If you only make changes to the layout on the diagram view in ODI, this will not be seen as a modification that needs to be committed.

When you commit, VCR4ODI will display the list of all objects that will be committed, indicating their change status: unchanged, added or changed.

Here also, VCR4ODI locks the objects in order to prevent others from working on those objects when the commit is running.

Before confirming to execute the commit operation, you can enter the requirement or issue numbers that have been solved with this commit. Note that this comment will be picked up by IKAN ALM to automatically create a link with the Issue Tracking System.



Entering the requirements or issue number and comment

IKAN ALM

Once the developers have committed their code into Subversion, IKAN ALM comes into play. The main objective of IKAN ALM is to help your organization to automate the ODI Life Cycle steps. By automating those steps you will make the process transparent, repeatable and documented. This will help you to master the related costs and risks and to enhance the quality, and it will contribute to the business by allowing for more frequent, bug-free, releases.

IKAN ALM provides the following web-based services:

- possibility to define a life cycle per project,
- define build and deploy processes,
- make build and deploy processes approval-based,
- update issue tracking systems,
- notify users regarding executed IKAN ALM requests.

IKAN ALM setup

There are several important steps in setting up a Project in IKAN ALM:

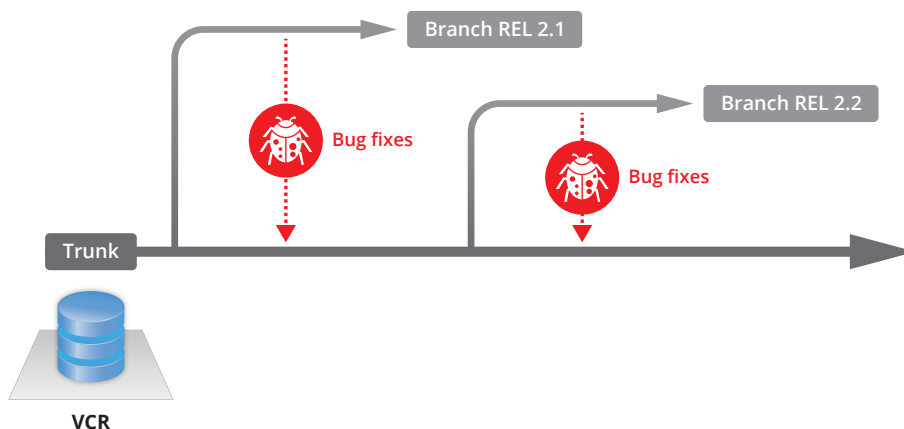
1. Defining the versioning strategy
2. Specifying the Life Cycle of your application
3. Setting up the Build and Deploy Environments
4. Using Phases and Phase Parameters
5. Specifying Environment Parameters

Step 1: Defining the Versioning Strategy

It is important to first define your versioning strategy.

A general accepted way is to use the Trunk (called Head in IKAN ALM) for ongoing development and to create a Branch for each Release. When fixes or enhancements are needed for Releases already in production, they can be done on that Branch and the related Life Cycle can be used for further deployment. When final, the changes at Release level must be merged at Trunk or Head level.

Throughout the remainder of this section, we will use the following sample versioning strategy:



It is the responsibility of the customer to define the Life Cycle of his product, i.e., how he wants to work. A classic example is DTAP: Development, Test , Acceptance (User Acceptance testing) and Production.

Project Administration>Life-Cycles Overview ?

Used for ongoing development

Life-Cycle : BASE

		Description	Base
		BASE Lifecycle	

Defined Levels

		Name	Description	Type	Locked	Optional	Notification Type (Criteria)	Requester	Pre-Notify	Pre-Approve	Post-Approve	Post-Notify (Criteria)
		BUILD	BUILD Level	Build			No notification (Never)					
		TESTING	Work Repo Tests with Create option	Test			No notification (Never)					
		UAT	Work Repo Acceptance Tests	Test			No notification (Never)					
		PRODUCTION	PRODUCTION Level	Production			No notification (Never)					

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

Used for Releases

Life-Cycle : RELEASE

		Description	Base
		LC for Releases	

Defined Levels

		Name	Description	Type	Locked	Optional	Notification Type (Criteria)	Requester	Pre-Notify	Pre-Approve	Post-Approve	Post-Notify (Criteria)
		BUILD	BUILD Level	Build			No notification (Never)					
		TESTING	Work Repo Tests with Create option	Test			No notification (Never)					
		UAT	Work Repo Acceptance Tests	Test			No notification (Never)					
		PRODUCTION	PRODUCTION Level	Production			No notification (Never)					

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

Trunk and Branches life cycles

There are three Level Types:

- **Build Level** must have at least one Build Environment
- **Test Level** must have at least one Build or Deploy Environment. A typical Test Level may have one or more Build Environments for Rebuilds and one or more Deploy Environments for Deployment and Testing
- **Production Level** must have at least one Build or Deploy Environment. A typical Production Level has one or more Deploy Environments.

Step 3: Setting up the Build and Deploy Environments

Next, you need to define your Build and Deploy environments. Basically, you have to specify the Machine, the Build/Deploy Tool and the Build/Deploy script you will use, and the Source and Target Locations.

Build Environments Overview										
	Name	Build Suffix	Machine	Source Location	Target Location	Build Script	Level	Build Tool	Downloadable Build	Debug
   	BUILD		ikandemo004v	E:/ikan/alm/webtest_environments/ODI_Master/CONTBUILD/BUILD/source	E:/ikan/alm/webtest_environments/ODI_Master/CONTBUILD/BUILD/target	build.xml	BUILD ANT1.8.1			

Example of a Build environment

Step 4: Using Phases and Phase Parameters

Phases are pre-defined, parameter-driven, packaged reusable scripts.

IKAN ALM provides basic CORE Phases, but you can also create your own Phases to customize the workflow of your projects, e.g., to upload your scripts, to define specific Parameters. Phases can be shared between different Projects and also between different IKAN ALM installations.

The benefits of creating a Phase to perform a task instead of running one large monolithic script are:

- ✓ **Reusability**
- ✓ **Improved logging:** by using many small Phases, the workflow is split up into smaller fractions which facilitates the understanding and monitoring of the workflow.
- ✓ **Versioning:** A Phase is identified by a unique name/version combination. When the scripts inside a Phase change, the version number will change as well.
- ✓ **Maintainability:** by using many small, limited-purpose Phases, the scripts being used tend to be smaller and easier to maintain.
- ✓ **Improved management of parameters**

More information on the IKAN ALM Phases concept can be found in the whitepaper "Using and Developing Custom Phases" at <http://www.ikanalm.com/infocenter/whitepapers.html>

Project Administration>Deploy Environment Phases Overview ?

Deploy Environment

Name PRODUCTION

Source Location E:/ikan/alm/webtest_environments/ ODI_Master/TEST/ TESTDEPLOY/source

Target Location E:/ikan/alm/webtest_environments/ ODI_Master/PROD/ PRODDEPLOY/target

Deploy Script

Partial Deploy No

Debug No

Level PRODUCTION

Machine ikandemo004v

Deploy Tool ANT1.8.1

Build Environment BUILD

Phases Overview

		Phase Name	Phase Version	Fail On Error	Next Phase On Error
↓	✖	Transport Build Result	5.6.0	Yes	Cleanup Build Result
↑	✖	Decompress Build Result	5.6.0	Yes	Cleanup Build Result
↑	✖	Copy Components with Filter	1.0.0	Yes	Cleanup Build Result
↑	✖	Oracle ODI Files Deployment	1.0.0	Yes	Cleanup Build Result
↑	✖	Cleanup Build Result	5.6.0	No	

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

Example of Deploy Environment Phases

In our example, a series of Parameters have been defined for the Oracle ODI Files Deployment Phase.

Global Administration>Edit Phase ?

Phase Info

Name com.ikanalm.phases.ant.scripting.deployODIFiles **Execution Type** ANT

Version 1.0.0 **Core Phase** No

Default Display Name Oracle ODI Files Deployment **Certified** No

Author IKAN **Released** No

Description Deploying Oracle ODI Files and/or Master or Work repositories creation.

Uploaded Files createODIRepository.xml
deployODI.xml
antext/lib/ant-contrib-1.0b3.jar
antext/lib/be.ikan.scm4all.client.ant.security.jar
antext/lib/com.springsource.org.apache.commons.codec.jar

[History](#) [Save](#) [Edit](#) [Refresh](#) [Overview](#) [Release](#) [Export](#) [Copy](#) [Upload](#)

Phase Parameters

	Name	Default value	Description	Mandatory	Secure	Integration Type
✖	confi.dir	\${target}/config				None
✖	project.codes		ODI Project name(s) to restore			None
✖	repodi.action	restoreAllObjects	ODI Repository Action to do: restoreAllObjects, restoreFromProjectList			None
✖	odi.passwd	*****	ODI Supervisor password		✓	None
✖	odi.userid	SUPERVISOR	ODI Supervisor name	✓		None
✖	odi.home	[ODI_HOME]	ODI_HOME Location (i.e.[ODI_HOME]/oracledi.sdk/lib/..)			None
✖	work.name	\${alm.project.name}	ODI Work repository name	✓		None
✖	work.sid	?SID?	ODI Work repository SID			None
✖	rdhmcWork host	localhost	DB Host of Work			None

[Create Parameter](#)

Deploy Phase parameters

Step 5: Specifying Environment Parameters

For each environment, you can also define Environment Parameters. In our example, a Parameter specifying the ODI Home variable has been defined for the Deploy Environment.

Project Administration > Parameters Overview

Build Environment

Deploy Environment

Parameter Type

Key

Machine

Show Machine Parameters

Mandatory

Editable

Dynamic

Secure

Search

Reset

Environment	Type	Machine	Actions	Key	Value	Description	Mandatory	Editable	Dynamic
PRODUCTION	Deploy	ikandemo004v		odi.home	C:/oracle/product/ODI_11117	ODI Product Path	✓		
				project.codes	PRJ_VCR4ODI			✓	
				repodi.action	restoreAllObjects;restoreFromProjectList				✓
				vcr.workcopy	\${target}/odi		✓		

Environment parameters overview

IKAN ALM Level Requests

Once the IKAN ALM setup is finished, you can start working. The different IKAN ALM activities are called level Requests. A Level Request can be initiated via the command line interface (CLI) or via the web interface. For the examples in this white paper we will use the web interface. A customizable Desktop displays the Project Streams you are working on. In our example, we have the ongoing development (BASE) and two Releases.

Desktop Overview								
Desktop Type	Project Stream [Package]	Level	Next Scheduled Request	Latest Level Request		Latest Successful Level Request	Action	Message
Project Stream	ODI_Master H_2.0	BUILD	9/19/14 3:18:00 PM	+ Release 2.0 b28[9/24/14 10:22:34 AM] ...		Release 2.0 b28[9/24/14 10:22:34 AM]	Level Request action buttons	✗
	ODI_Master H_2.0	TESTING		+ Release 2.0 b28[9/24/14 12:42:28 PM] ...		Release 2.0 b28[9/24/14 12:42:28 PM]		
	ODI_Master H_2.0	UAT		+ Release 2.0 b28[9/24/14 10:49:50 AM] ...		Release 2.0 b28[9/24/14 10:49:50 AM]		
	ODI_Master H_2.0	PRODUCTION		+ Release 2.0 b28[9/24/14 10:53:37 AM] ...		Release 2.0 b28[9/24/14 10:53:37 AM]		
Project Stream	ODI_Master B_2-2	BUILD	9/19/14 3:18:00 PM	+ Release 2.2 b4[9/22/14 6:16:47 PM] ...		Release 2.2 b4[9/22/14 6:16:47 PM]	Level Request action buttons	✗
	ODI_Master B_2-2	TESTING		+ Release 2.2 b4[9/22/14 6:51:50 PM] ...		Release 2.2 b4[9/22/14 6:51:50 PM]		
	ODI_Master B_2-2	UAT						
	ODI_Master B_2-2	PRODUCTION						
Project Stream	ODI_Master B_2-1	BUILD	9/19/14 3:18:00 PM	+ Release 2.1 b3[9/19/14 12:28:10 PM] ...		Release 2.1 b3[9/19/14 12:28:10 PM]	Level Request action buttons	✗
	ODI_Master B_2-1	TESTING						
	ODI_Master B_2-1	UAT						

Desktop displaying the Level Requests

After having selected the appropriate Level Request action button, the Level Request will be executed and you are forwarded to the Level Request Detail screen which displays the actual status and other relevant information concerning the current Level Request.

This screen is a starting point for consulting even more detailed information using the links and tab pages provided.

Desktop>Level Request Detail ?

Status of the Level Request and links to the related Project, Release, Level and Build/Deploy information

Success [ODI_Master](#) / [H_2.0](#) / [PRODUCTION](#) / [Build# 28](#)

99: test
Requested by: global on: 9/24/14 10:50:07 AM

Different tabs leading to more detailed information concerning the Level Request.

Summary | **Phase Logs** | Results | Approvals | Issues | Sources | Modifications | Dependencies

[Back](#) | [Refresh](#)

Actions

No actions available

Info

Build Number 28
VCR Tag Release_2.0_b28
Action Deliver Build
Type Deploys of archived Build
Start 9/24/14 10:50:07 AM
Duration 00:03:30
[Show more...](#)

Builds & Deploys

	OID	Environment	Machine	Start	Duration
	48	PRODUCTION	ikandemo004v	9/24/14 10:50:07 AM	00:03:30

Issues

[Link to the issue in the Issue Tracking System.](#)

Issue ID	Description	Status	Owner	Priority
ODI-1	This is just for demo purposes	Open	global	Major

Level Requests detailed overview

For more detailed information, we refer to the IKAN ALM User Guide which you can find at <http://www.ikanalm.com/infocenter/documentation.html>

Conclusion

RedBridge Software's Life Cycle Management solution adds value to ODI by combining ODI, Subversion, VCR4ODI and IKAN ALM. It provides the missing link to achieve real, automated, Life Cycle Management for ODI.

The main benefits of our solution are:

- ✓ Proper storage and versioning of all software objects
- ✓ Cost reduction in both development, testing and deployment
- ✓ Tasks are automated and repeatable
- ✓ Reduced risk by automated, repeatable build and deploy processes
- ✓ Improved application quality
- ✓ Faster time-to-market
- ✓ Less resources and time needed for build and deploy to test and production
- ✓ Better communication between stakeholders
- ✓ Better visibility over the whole process
- ✓ Well defined and automated development and deploy process

On the RedBridge Software website you can also find a white paper on Life Cycle Management and a tool to calculate the expected Return on Investment when implementing our LCM for ODI Solution.

For more information

To know more, visit <http://www.redbridgesoftware.com>
Contact RedBridge Software: info@redbridgesoftware.com



RedBridge Software (Belgium)

Kardinaal Mercierplein 2
2800 Mechelen, Belgium
Tel. +32 15 44 50 44

RedBridge Software (France)

3, Rue du Général De Gaulle
28700 Aunay-Sous-Auneau, France
Tél: +33 2 37 25 31 22

info@redbridgesoftware.com
www.redbridgesoftware.com

© Copyright 2019 RedBridge Software BVBA.

The RedBridge Software and RedBridge logos and names and all other RedBridge product or service names are trademarks of RedBridge Software BVBA. 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 RedBridge Software BVBA.