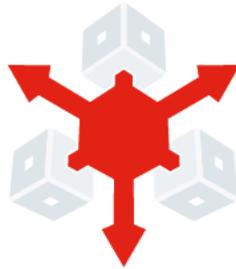


# Getting Started Guide

Release 5.8

January 2019



# IKANALM

IKAN Development N.V.  
Kardinaal Mercierplein 2  
2800 Mechelen  
BELGIUM

© 2006 - 2019 IKAN Development N.V.

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.

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.

---

# Table of Contents

<b>Chapter 1 - Purpose .....</b>	<b>1</b>
<b>Chapter 2 - Prerequisites.....</b>	<b>2</b>
2.1. IKAN ALM Installation .....	2
2.2. Subversion Versioning System.....	2
2.3. IKAN ALM Installation Information.....	3
<b>Chapter 3 - Global Administration: Defining Subversion and Creating a Project</b>	<b>4</b>
3.1. Global Administration: Initial Overview.....	4
3.2. Subversion Setup and Definition.....	5
3.3. Creating the Docgen Project.....	7
<b>Chapter 4 - Setting up the Build Level: Generating HTML and PDF.....</b>	<b>9</b>
4.1. Creating the Build Level.....	9
4.2. Creating the Build Environment.....	10
4.3. Auditing the Project .....	10
4.4. Creating the Build Level Request.....	11
4.5. Verifying the Build Level Request.....	12
<b>Chapter 5 - Setting up the Test Level: Deploy to the Web Server .....</b>	<b>16</b>
5.1. Creating the Test Level .....	16
5.2. Creating the Deploy Environment.....	16
5.3. Creating the Deploy Parameters for Authentication on the Web Server Manager Application .....	17
5.4. Auditing the Project .....	19
5.5. Creating the Deliver Level Request.....	19
5.6. Verifying the Deliver Level Request .....	20
<b>Chapter 6 - Phases.....</b>	<b>23</b>
6.1. Global Administration: Importing Phases.....	23
6.2. Build Environment: Inserting and Configuring Phases.....	24
6.3. Creating and verifying the Build Level Request.....	27
<b>Chapter 7 - Additional Information.....</b>	<b>29</b>

# CHAPTER 1

## Purpose

This *Getting Started Guide* explains how to set up a very basic Project, called *Docgen*, in IKAN ALM, using Build and Deploy tasks. It contains an introduction to the most important IKAN ALM concepts. The target audience are ALM engineers who want to know how to configure the IKAN ALM Global Administration and how to create and manage a Project that builds sources and deploys them to a Test or Production server.

We assume that you have some basic DevOps or Application Lifecycle Management knowledge, including working with a versioning system and creating (build or deploy) scripts. We also assume that you already had a look at the IKAN ALM demo installation (note that it is possible to configure the *Docgen* Project starting from the Demo installation) and have grasped the basic stuff, such as logging on and navigating through the Global Administration, Project Administration and Desktop sections within IKAN ALM.

During the Build process, the sample *Docgen* Project will convert a standardized XML document into a PDF and an HTML file and, next, deploy them to a web server.

We will use the [Apache ANT](#) scripting tool and the [Apache Tomcat](#) web server, but no prior knowledge of either tool is necessary since the samples are kept very basic.

The source XML file is very simple and follows the [docbook](#) format. You will find it in the *GettingStarted.zip* file under `/init_svn/sources/docbook.xml`.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <article xml:lang="en" xmlns:xi="http://www.w3.org/2001/XInclude">
3
4 <title>Docbook article as an IKAN ALM showcase</title>
5 <section id="section1">
6 <title>Phases</title>
7 <para>
8 When IKAN ALM is running Level Requests, Builds and Deploys, all actions are
9 performed by executing a sequence of Phases. Those Phases are defined in the
10 IKAN ALM database and can be consulted and manipulated in the Phases section of
11 the Global Administration interface. Once they have been defined in Global
12 Administration, Phases may be linked to Levels, Build or Deploy Environments in
13 the Project Administration context.</para>
14
15 <para>
16 The IKAN ALM core functionality is performed by so-called "Core" Phases.
17 Those Core Phases can only be viewed, and cannot be altered nor deleted.
18 Consider them an integral part of IKAN ALM.</para>
19
20 <para>You can extend this core functionality by creating your own Phases.
21 There are two options:</para>
22
23 <orderedlist numeration = "arabic">
24 <listitem>
25 <para>Create a Phase from scratch using the "Create Phase" functionality.
26 In that case, you first specify the name and the version of the Phase, and,
27 next, you choose the script or scripts to be executed by the Phase.
28 </para>
29 </listitem>
30 <listitem>
31 <para>Import a Phase that has already been created via the "Import Phase"
32 functionality.
33 </para>
34 </listitem>
35 </orderedlist>
36 </section>
```

---

# Prerequisites

## 2.1. IKAN ALM Installation

- A correctly installed IKAN ALM Server 5.8, with the web application preferably being deployed to Apache Tomcat. The Demo project contains a Test level, whereby results will be deployed to the Tomcat web server running the IKAN ALM web application.
- A correctly installed IKAN ALM Agent 5.8, with access to an [Apache ANT](#) scripting tool (minimum version 1.8.2). The recommended way is to install that IKAN ALM Agent on the same Machine as the IKAN ALM Server since it has an integrated and configured ANT definition (See [Global Administration: Defining Subversion and Creating a Project](#) on page 4.). If that Agent on the Server machine does not fit into your ALM architecture, you may always disable it later on by removing the service (Windows) or daemon process (Linux/Unix).
- Tomcat version 7 or higher is required.

---

**Note:** The IKAN ALM Demo installation will also fulfill the following requirements: the IKAN ALM Server and Agent will be installed and the Ant scripting tool and Subversion repository will be installed and configured. On Windows, it also includes the Subversion server; on Linux, you have to install Subversion separately.

---

## 2.2. Subversion Versioning System

This *Getting Started Guide* includes the creation of a Subversion repository. In order to connect to this repository, you need Subversion version 1.6 or higher installed on the IKAN ALM Server. The [Apache Subversion website](#) provides links for downloading a binary subversion package for your system.

For example, on a Windows operating system you can easily install the Win32Svn package. If you download the current (November 2017) [svn-win32-1.8.16.zip](#) archive, the installation is as easy as an unzip action.

The installation on Linux is mostly done through a Package manager. For example, on Suse Linux, you can do this using zypper:

```
$ zypper install subversion
$zypper install subversion-server
```

For more information on how to work with Subversion, refer to the *Subversion Quick Start Guide* at <http://svnbook.red-bean.com/en/1.7/svn.intro.html>.

## 2.3. IKAN ALM Installation Information

The following additional information is needed to complete the procedures explained in this *Getting Started Guide*:

- *ALM\_HOME*

This is the installation folder of the IKAN ALM Server. You can derive this location from the Local File Copy Locations under *Global Administration > System Settings*.

For example: if the Work Copy Location is set to `E:/ALM/system/workCopy`, `ALM_HOME = E:/ALM`.

- *ANT\_HOME*

Normally Ant is installed in the `ALM_HOME` folder. If not, the exact location can be found in the field *Java ANT Classpath* under *Global Administration > Scripting Tools > Overview ANT Scripting Tools*.

For example: if the Java ANT Classpath is set to `E:/ALM/ant/lib/ant-launcher.jar; ...`, `ANT_HOME=E:/ALM/ant`.

- *JAVA\_HOME*

Location of the Java Runtime on the IKAN ALM Server.

For example: the location provided during the installation procedure or the `JAVA_HOME` location of the ANT tool as mentioned above.

- *TOMCAT\_HOME*

Root location of the Apache Tomcat web server that is hosting the IKAN ALM web application.

If you are using the Demo version, the root location will be `ALM_HOME/appServer`.

For example: the location provided during the installation of IKAN ALM.

- ALM URL Elements

*tomcatHost* and *tomcatPort*: the ALM webapplication is accessible through a URL which consists of several elements: `http://tomcatHost:tomcatPort/alm`. You can retrieve the elements from the IKAN ALM URL which you can find on the *Mail* tab under *Global Administration > System Settings*.

For example `http://localhost:9080/alm` with `tomcatHost=localhost` and `tomcatPort=8080`.

**Note:** 9080 is the standard port defined for the Demo installation, 8080 is the standard *tomcatPort*.

- For the deployment of our application to the web server, we need some information about the Tomcat setup, so that we can stop and start it. Check the documentation on the [Tomcat website](#), for more information on how to do that.

---

**Note:** If you did not install IKAN ALM yourself, you should ask your system manager to provide you with the correct installation paths and variables.

---

# Global Administration: Defining Subversion and Creating a Project

So far, so good. Now, let's dive into IKAN ALM.

First, we will:

- verify what is already preconfigured in the Global Administration section,
- initialize and connect to the Subversion repository,
- and create our first project.

## 3.1. Global Administration: Initial Overview

Let's start with verifying what is already set up in the IKAN ALM Global Administration after a clean installation. We will describe it shortly without going into detail as, after all, we are eager to create our own project. If however, you want to know more about a specific topic, have a look at the respective chapters in the *Global Administration* part of the *IKAN ALM User Guide*.

Log on to IKAN ALM with user *global* and password *global*, and select *Global Administration* from the Main Menu. Via this starting place (or via one of the submenus) you can verify the following settings:

1. Under *System > System Settings*, you will see the *Build Archive Location* on the IKAN ALM Server, where all the Build Artifacts (e.g., PDFs, deployable archives, ...) will be stored after a successful build, so that they can be deployed later in the lifecycle. It is a local path on the server, something like `C:/ALM/system/buildArchive`, or `/opt/alm/system/buildArchive`.
2. Under *Machines > Overview*, you will find the definition of the IKAN ALM Server. You should also have an Agent installed on this Machine, and both Agent and Server processes should be running.

Verify this by clicking the  *Installed Phases* link on the overview: you will be forwarded to the *Installed Phases Overview* which displays a column for the *Current Server Activity* and one for the *Current Agent Activity*, which both should have a green  icon as status. If the icon is red , verify the topic on how to *Start the IKAN ALM Agent/Server* in the respective *Agent* or *Server Installation Guide*.

3. Under *Scripting Tools > Overview ANT Scripting Tools*, an *ANT 1.9.3* scripting tool is defined pointing to the installed Ant version on the IKAN ALM server. We will use that Ant scripting tool to generate a PDF and an HTML file from the docbook XML, and to deploy those results to a web server.

After having verified those Global Administration settings, we are now almost ready to create a Project and start a Build. Almost ... as we of course need some sources to build. We will retrieve them from a Subversion repository that we will create and define in the next section.

## 3.2. Subversion Setup and Definition

The archive delivered with this *Getting Started Guide* contains scripts and sources to create a Subversion repository to which we will connect using the file protocol. As specified in the requirements, we assume you have Subversion installed on the IKAN ALM server. Extract the *GettingStarted.zip* in a temporary directory in the IKAN ALM installation folder.

Adapt the JAVA\_HOME and ANT\_HOME in the file *init\_svn/init\_svn\_repo.cmd* (Windows) or *init\_svn/init\_svn\_repo.cmd/sh* (Linux). Point the JAVA\_HOME to the one used for IKAN ALM, and use ANT that is installed with the IKAN ALM Server under ALM\_HOME/ant.

Sample file:

```
@attrib *.* -R /S /D

REM script that calls the ANT script init_svn_repo.xml :
REM - deletes and re-creates a Subversion repository
REM - imports the sources of the Web Test Projects into the Subversion repository

set JAVA_HOME=D:\java\jdk1.8
set ANT_HOME=E:\ALM\ant

@echo JAVA_HOME = %JAVA_HOME%
@SET CLASSPATH=%ANT_HOME%/lib/ant-launcher.jar
@SET ANTCMD=%JAVA_HOME%/bin/java -Dant.home=%ANT_HOME% -Xmx256m -cp %CLASSPATH%
org.apache.tools.ant.launch.Launcher

%ANTCMD% -f init_svn_repo.xml
```

Adapt the repository location (path and svn url, make sure that you have write access) and the svn bin path in the *init\_svn/init\_svn\_repo.properties* file.

Sample properties file:

```
# properties file used by init_svn_repo.xml

# if true, the repository is on the local machine and will be deleted and re-created
disLocalRepository=true
# path to the Subversion binaries
svn.bin.path=D:/vcrs/server/svn/1.8.5/bin
# local path to the Subversion repository
svn.repo.local.path=E:/ALM/GettingStarted/Docgen/repository
# remote URL of the Subversion repository
svn.repo.remote.URL=file:///E:/ALM/GettingStarted/Docgen/repository
```

---

**Note:** Do not use backward slashes.

---

Remember the *svn.bin.path* and *svn.repo.remote.URL* properties as we will need them when defining the Subversion repository in IKAN ALM.

Run *init\_svn\_repo.cmd* (Windows) or *init\_svn\_repo.sh* (Linux):

```

C:\Windows\system32\cmd.exe
E:\ALM\GettingStarted\init_svn>init_svn_repo.cmd
E:\ALM\GettingStarted\init_svn>REM script that calls the ANTI script init_svn_repo.xml :
E:\ALM\GettingStarted\init_svn>REM      - deletes and re-creates a Subversion repository
E:\ALM\GettingStarted\init_svn>REM      - imports the sources of the Web Test Projects into the Subversion repository
E:\ALM\GettingStarted\init_svn>set JAVA_HOME=D:\java\jdk1.7.0_21
E:\ALM\GettingStarted\init_svn>set ANT_HOME=E:\ALM\ant
JAVA_HOME = D:\java\jdk1.7.0_21
E:\ALM\GettingStarted\init_svn>D:\java\jdk1.7.0_21\bin/java -Dant.home=E:\ALM\ant -Xmx256m -cp E:\ALM\ant\lib\ant-launcher.jar org.apache.tools.ant.launch.Launcher -f init_svn_repo.xml
Buildfile: E:\ALM\GettingStarted\init_svn\init_svn_repo.xml

deleteRepository:

createRepository:
[mkdir] Created dir: E:\alm\GettingStarted\docgen\repository
[echo] Created SUN Repository at E:\alm\GettingStarted\docgen\repository

import_docgen_sources:
[mkdir] Created dir: E:\ALM\GettingStarted\init_svn\tmp
[mkdir] Created dir: E:\ALM\GettingStarted\init_svn\tmp\import\docgen\trunk
[mkdir] Created dir: E:\ALM\GettingStarted\init_svn\tmp\import\docgen\branches

es
[mkdir] Created dir: E:\ALM\GettingStarted\init_svn\tmp\import\docgen\tags
[copy] Copying 5 files to E:\ALM\GettingStarted\init_svn\tmp\import\docgen\tags

trunk
[exec] Adding      docgen
[exec] Adding      docgen\trunk
[exec] Adding      docgen\trunk\docbook.xml
[exec] Adding      docgen\trunk\lib
[exec] Adding      <bin> docgen\trunk\lib\ant4docbook-0.5.0.jar
[exec] Adding      docgen\trunk\deploy.xml
[exec] Adding      docgen\trunk\build.xml
[exec] Adding      docgen\trunk\wehapp
[exec] Adding      docgen\trunk\wehapp\WEB-INF
[exec] Adding      docgen\trunk\wehapp\WEB-INF\web.xml
[exec] Adding      docgen\branches
[exec] Adding      docgen\tags
[exec]
[exec] Committed revision 1.
[delete] Deleting directory E:\ALM\GettingStarted\init_svn\tmp

all:
BUILD SUCCESSFUL
Total time: 9 seconds

```

Now that the repository has been created, we need to define it in IKAN ALM In the *Global Administration* context, select *Version Control Repositories > Create Repository*.

The following screen is displayed:

Select *Subversion* from the drop-down list in the *Type* field. The *Connection Details* panel will be displayed.

Set the *Command Path* and *Repository URL* to what was set in the properties file (svn.bin.path and svn.repo.remote.url). The *User ID* and *Password* are not needed since we connect using a file URL. Set the *Repository Layout* to *Project-oriented*, and the *Time-Out* to 30 seconds.

Global Administration > Create Subversion Repository ?

INFO: COULD SUCCESSFULLY ESTABLISH A CONNECTION WITH THE REPOSITORY

### Create Subversion Repository

Type	Subversion
Name	SVNDemo
Description	Subversion repository containing the Docbook project.

### Subversion Connection Details

Command Path	C:/ALM/vcr/subversion/server/bin
User ID	
Password	
Repeat Password	
Repository URL	file:///E:/alm/GettingStarted/Docgen/repository
Tags Directory	tags
Trunk Directory	trunk
Repository Layout	Project-oriented
Time-Out (Sec.)	30

Fetch Meta Properties  Yes  No

Test Connection  
Create Reset

---

**Note:** You can first click the *Test Connection* button before you *Create* the repository.

---

### 3.3. Creating the Docgen Project

Now we can start with the actual creation of our Release-based Project.

In the *Global Administration* context, select *Project > Create Project* and fill out the fields as required.

Set the Project Type to Release-based. Specify the SVN repository containing the *Docgen* project and its project name in the VCR (Docgen). Choose *ANT* for the Build and Deploy Tool Type. Together with the Project, a Head Project Stream is created that points to the trunk in the *Docgen* project in Subversion. Provide a Build Prefix, make sure to set the Build Type to *Full Build* and to set Accept Forced Build to *Yes*.

**Note:** The name of the project in the SVN repository is case-sensitive!

Global Administration > Create Project ?

### Create Project

Project Settings	
<b>Project Type</b>	Release-based
<b>Name</b>	Docgen *
<b>Description</b>	Getting Started Sample project. Generates PDF and HTML documentation from a docbook XML file and deploys them onto a web server. *
<b>VCR</b>	SVNDemo *
<b>VCR Project Name</b>	Docgen *
Issue Tracking System	
<b>Build Tool Type</b>	ANT *
<b>Deploy Tool Type</b>	ANT *
<b>Build Script</b>	
<b>Deploy Script</b>	
<b>Locked</b>	Yes
<b>Hidden</b>	No
Project Security Settings (optional)	
<b>User Access</b>	
<b>Admin Access</b>	

Head Project Stream Settings	
<b>Prefix</b>	1 *
<b>Status</b>	Development
<b>Description</b>	Project Stream pointing to the trunk (master) in Docgen in Subversion
<b>Locked</b>	<input type="radio"/> Yes <input checked="" type="radio"/> No
<b>Hidden</b>	No
<b>Tag-Based</b>	<input type="radio"/> Yes <input checked="" type="radio"/> No
<b>Build Type</b>	Full Build
<b>Accept Forced Build</b>	<input checked="" type="radio"/> Yes <input type="radio"/> No
<b>Tag Template</b>	\${streamType}_\${prefix}_b\${buildNumber}
<b>VCR Branch ID</b>	

**Note:** You can first use the *Check Project Name in the VCR* button before you *Create the Project*.

# Setting up the Build Level: Generating HTML and PDF

Let's do some work in our new Project, so that we can “build” the documentation!

## 4.1. Creating the Build Level

Under *Lifecycles > Overview* you will notice that the BASE Lifecycle has been created which is linked to the HEAD Project Stream. Select the *Edit* link next to this Lifecycle.

To be able to build, we first have to create a first step (Level) in the Lifecycle by selecting the *Create Build Level* link underneath the empty table of *Defined Levels*.

The screenshot shows the 'Create BUILD Level' form in the Project Administration interface. The form is titled 'Create BUILD Level' and contains the following fields and options:

- Name:** BUILD
- Description:** "Compile" docbook.xml in PDF and HTML. Generate a deployable web archive (war)
- Type:** Build
- Locked:** Yes
- Debug:** Yes (selected), No
- Notification Type:** No notification
- Notification Criteria:** Never
- Schedule:** (empty dropdown)
- Requester User Group:** (empty dropdown)
- Life-Cycle:** BASE

At the bottom of the form are three buttons: Create, Reset, and Back.

Most fields speak for themselves (let's neglect the Notification, Schedule and Requester fields for now). Activating the *Debug* option makes it easier to track stuff in the beginning, especially when a Build fails. Once everything runs smoothly, set it to *No*. Together with the Level, the Phases linked to the Level are created. Those Phases will be executed when a Request will be executed on the IKAN ALM Server (see later).

You can check those Phases by selecting the  *Edit Phases* link underneath the Build Level under *Levels > Overview*.

## 4.2. Creating the Build Environment

A Level is a conceptual step in the Lifecycle. As we need a concrete Machine to execute our Build on, we need to link a Build Environment to the Level.

Select *Create* on the *Build Environments* submenu.

The screenshot shows the 'Create Build Environment' dialog box. It contains the following fields and values:

- Name: build
- Level: BUILD
- Machine: ikan521v
- Build Tool: ANT1.9.3
- Source Location: E:/ALM/env/docgen/BUILD/build/source
- Target Location: E:/ALM/env/docgen/BUILD/build/target
- Build Suffix: (empty)
- Downloadable Build: Yes
- Debug: Yes

Buttons for 'Create' and 'Reset' are visible at the bottom.

The Build will be executed by the IKAN ALM Agent on the *Machine* that we will select, and more precisely by the selected (1.9.3) *Ant* Scripting Tool. The defined *build.xml* is an Ant script that you can inspect in the sources under `/init_svn/source` and that is imported in the Subversion trunk of the *Docgen* project. The script is not very complex: it will execute four Ant targets: (1) generate an HTML file and (2) generate a PDF file based on the *docbook.xml*, (3) copy the resulting files to the target location so that they will end up in the build result and finally (4) generate a deployable web archive in the target that we can deploy (see later).

The sources will be transferred to a subdirectory of the *Source Location*. The result must be placed in the *Target Location*. Note that, normally, those locations will be cleaned up after the Build, unless you activate the *Debug* option, what we do for the same reason as explained for the Build Level.

**Note:** The source and target location are preferably located in the ALM\_HOME installation folder. You are free to specify the location of those folders. The structure will be automatically created.

In our example we use `E:/ALM/env/docgen/BUILD/build/source`.

`E:/ALM` is the ALM\_HOME installation directory followed by the `env` directory containing our projects, `docgen` (the project directory), `BUILD` (the level), `build` (the environment) and finally the source or target directory.

In order to distinguish Levels from Environments, we use uppercase for the level and lowercase for the environment directories.

Set *Downloadable Build* to *Yes*, so that we can download and check the build result.

Just as for the Level, the Phases linked to the Environment are created together with the Build Environment. They will be executed when the Build of a Level Request will be executed on the IKAN ALM Agent (see later). You can check those Phases by selecting the  *Edit Phases* link next to the Build Environment under *Build Environments > Overview*.

## 4.3. Auditing the Project

Everything is ready to start a Build, except that the Level was locked when we created it: first we need to verify if our definitions are consistent. Do so by selecting *Audit Project* on the Main Menu.

On the overview, you will see most of the different objects we created.

The information screen for the *Docgen* Project displays the Build Archive of the Head Project Stream (where our future Builds will be stored) and the Build Level containing one Build Environment on the IKAN ALM Agent, where the *build.xml* script will be executed by an Ant scripting tool.

Click the *Unlock* link, and we are ready to build!

## 4.4. Creating the Build Level Request

First we will add the Head Project Stream of the *Docgen* Project to our Desktop. Go to your Desktop and click the *Add to Desktop* button at the bottom of the screen. In the pop-up window, find the Head Project Stream (1) of the *Docgen* project, select it and click *Add to Desktop*.

Desktop > Desktop  Auto Refresh

1 +

Desktop Overview							
Desktop Type	Project Stream [Package]	Level	Next Scheduled Request	Latest Level Request	Latest Successful Level Request	Action	Message
Project Stream	Docgen H_1	BUILD					

Click the *Request icon* in the *Action* column of our Project Stream.

Desktop > Create Level Request

**Request Build** Docgen / H\_1 / BUILD  
Getting started sample project.  
Active Build Number: 0

[Back](#) | [Show Additional Info](#) | [Hide Modifications](#)

**Modifications since previous successful Level Request**

			Path	User	Modification Date
			.	changeadmin	30/01/18 16:34:18
			build.xml	changeadmin	30/01/18 16:34:18
			deploy.xml	changeadmin	30/01/18 16:34:18
			docbook.xml	changeadmin	30/01/18 16:34:18
			lib	changeadmin	30/01/18 16:34:18
			lib/ant4docbook-0.5.0.jar	changeadmin	30/01/18 16:34:18
			webapp	changeadmin	30/01/18 16:34:18
			webapp/WEB-INF	changeadmin	30/01/18 16:34:18
			webapp/WEB-INF/web.xml	changeadmin	30/01/18 16:34:18

9 items found, displaying all

**Description** Build Level Request \*

**Previous Descriptions**

**Build Number** 1

**VCR Tag** H\_1\_b1

Click the *Show Additional Info* link.

Notice that the Build Environment is linked to the Level. Provide a meaningful description, do not modify the Indicative VCR Tag that will be created in Subversion when the Build is successful, click the *Show Modifications* link and have a look at all the sources we have imported in `Docgen/trunk` (our `docbook.xml`, the build and deploy script, ...) when creating the Subversion repository.

Finally, select the *Create* button.

## 4.5. Verifying the Build Level Request

You are forwarded to the *Desktop* on which the Request you created is displayed. Click through on its *link* (H\_1\_b1 if you started on a clean installation) to see what has happened during the Build.

### Level Request Detail

The header of the *Level Request Detail* screen displays information on the status of the Level Request. The different tab pages underneath the header display additional information such as the status of each of the Phases (on the *Phase Logs* tab).

This is how it should look when the Request is finished:

The screenshot shows the 'Level Request Detail' page for a build. At the top, there is a warning icon and a message: 'Warning Docgen / H\_1 / BUILD / Build# 1: 1: First Build: generate the documentation Requested by: global on: 7/31/14 9:32:11 AM'. Below this is a red instruction: 'Select one of the tab pages for additional information.' A row of tabs is visible: 'Summary', 'Phase Logs', 'Results', 'Approvals', 'Issues', 'Sources', 'Modifications', and 'Dependencies'. The 'Phase Logs' tab is selected. Below the tabs are three main sections: 'Actions' (No actions available), 'Info' (Build Number 1, VCR Tag H\_1\_b1, Action Request Build, Type Builds based on latest code, Start 7/31/14 9:32:12 AM, Duration 00:00:21, Show more...), and 'Builds & Deploys' (a table with one row: + [icon] 1 build ikan028 7/31/14 9:32:16 AM 00:00:16). At the bottom, there is a 'Warning log' section with a warning icon, showing a phase 'Cleanup Source' starting at 7/31/14 9:32:33 AM with a duration of < 1 sec. and a status of 'Warning'. A message box below contains the text: 'Source Location E:/ALM/env/docgen/BUILD/build/source/1 not cleaned up, because of enabled debugging on the environment'.

Select the *Results* tab page. This page shows the result of the Build executed by the IKAN ALM Agent on the Build Environment.

Desktop>Results ?

**Warning** Docgen / H\_1 / BUILD / Build# 1  
**1: First Build: generate the documentation**  
 Requested by: global on: 7/31/14 9:32:11 AM

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

← Back | Refresh

**Results**

▼ Build: 1

<b>Build File Name</b> Docgen_H_1_b1_BUILD.zip	<b>Environment</b> build
<b>File Size</b> 40 KB	<b>Machine</b> ikan028
<b>Archive Status</b> Present	<b>Status</b> +

[Download Build Result](#)

Docgen\_H\_1\_b1\_BUILD.zip

Click the *Download Build Result* link to download and check the Build Result.

Ann Oversteins > AppData > Local > Temp > Docgen\_H\_1\_b1\_BUILD.zip

Name	Type	Compressed size	Password ...	Size
css	File folder			
deploy.xml	XML File	2 KB	No	
docbook.pdf	Adobe Acrobat Document	11 KB	No	
docgen.war	WAR File	20 KB	No	
index.html	Firefox HTML Document	3 KB	No	

docbook.pdf - Adobe Reader

File Edit View Window Help

1 (1 of 2) 45,3% Tools Sign Comment

**Docbook article as an IKAN ALM showcase**

**Table of Contents**

Phases	1
Creating a Phase Definition	1
More information	2

**Phases**

When IKAN ALM is running Level Requests, Builds and Deploys, all actions are performed by executing a sequence of Phases. Those Phases are defined in the IKAN ALM database and can be consulted and manipulated in the Phases section of the Global Administration interface. Once they have been defined in Global Administration, Phases may be linked to Levels, Build or Deploy Environments in the Project Administration context.

The IKAN ALM core functionality is performed by so-called "Core" Phases. Those Core Phases can only be viewed, and cannot be altered nor deleted. Consider them an integral part of IKAN ALM.

You can extend this core functionality by creating your own Phases. There are two options:

1. Create a Phase from scratch using the "Create Phase" functionality. In that case, you first specify the name and the version of the Phase, and, next, you choose the script or scripts to be executed by the Phase.
2. Import a Phase that has already been created via the "Import Phase" functionality.

**Creating a Phase Definition**

Compressed size: 10,0 KB  
 Size: 13,0 KB  
 Ratio: 23%

Open *docbook.pdf/index.html* to check if the conversion to PDF/HTML was successful. Also, have a look at the *docgen.war* that was created. That web archive will be used later on.

## Build Phases Log

The different Build Phases form the workflow of a build. They are automatically created together with the Build environment.

Select the *Phase Logs* tab page. The grey lines on the overview represent the different Phases, the white lines represent the Build or Deploy actions.

Desktop>Phase Logs ?

**Warning** Docgen / H\_1 / BUILD / Build# 1  
1: First Build: generate the documentation  
Requested by: global on: 7/31/14 9:32:11 AM

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

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

### Phase Logs

Phase Name	Start Date/Time	Duration
Retrieve Code	7/31/14 9:32:12 AM	00:00:03
Build	7/31/14 9:32:16 AM	00:00:16
<b>Build 1 on machine ikan028</b>	7/31/14 9:32:16 AM	00:00:16
Tag Code	7/31/14 9:32:33 AM	< 1 sec.
Deploy	7/31/14 9:32:33 AM	< 1 sec.
Cleanup Work Copy	7/31/14 9:32:33 AM	< 1 sec.

Click the Build name (in our example: *Build 1 on machine ikan028*) to expand the information panel displaying the Phase details.

**Build 1 on machine ikan028** 7/31/14 9:32:16 AM 00:00:16

OID 1 Start Date/Time 7/31/14 9:32:16 AM  
Environment build Duration 00:00:16  
Machine ikan028 Status Warning

#### Build Parameters

Transport Source	7/31/14 9:32:19 AM	< 1 sec.
Verify Build Script	7/31/14 9:32:19 AM	< 1 sec.
Execute Script	7/31/14 9:32:19 AM	00:00:13
Transport Deploy Script	7/31/14 9:32:33 AM	< 1 sec.
Compress Build	7/31/14 9:32:33 AM	< 1 sec.
Archive Result	7/31/14 9:32:33 AM	< 1 sec.
Cleanup Source	7/31/14 9:32:33 AM	< 1 sec.
Cleanup Result	7/31/14 9:32:33 AM	< 1 sec.

Next, you can click one of the Phases to immediately jump to its *Phase Log*.

The *Execute Script* Phase Log shows the result of the build.xml script executed by the Ant scripting tool.

**Execute Script** 7/31/14 9:32:19 AM 00:00:13

**Phase Name** Execute Script - 5.5.0 **Duration** 00:00:13  
**Start Date/Time** 7/31/14 9:32:19 AM **Status** Success

**Message**

**Log**

[Download Log](#)

```

generate_html:
[dbk] process done : E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom
[dbk] use parameter : html.stylesheet = css\jbossorg.css
[dbk] COPY C:\Users\ano\ant4docbook\0.5.0\css TO E:\ALM\env\docgen\BUILD\build\source\1\Docgen
[dbk-xslt] Processing E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom to E:\ALM\env\docgen\BUILD\build\source\1\Docgen\index.html
[dbk-xslt] Loading stylesheet C:\Users\ano\ant4docbook\docbook-xsl-1.77.1\html\docbook.xsl
[dbk] DELETING TEMP FILES... USE property '<property name="ant4docbook.keepTempFiles" value="true"/>' to keep them
[dbk] DELETE FILE : E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom
[dbk] process done : E:\ALM\env\docgen\BUILD\build\source\1\Docgen\index.html

generate_pdf:
[dbk] process done : E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom
[dbk-xslt] Processing E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom to E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.x
[dbk-xslt] Loading stylesheet C:\Users\ano\ant4docbook\docbook-xsl-1.77.1\fo\docbook.xsl
[dbk-xslt] C:\Users\ano\ant4docbook\docbook-xsl-1.77.1\fo\docbook.xsl:320:16: Warning! Making portrait pages on USletter paper (8.5inx11in)
[dbk] juil. 31, 2014 9:32:32 AM org.apache.fop.events.LoggingEventListener processEvent
[dbk] WARNING: The following feature isn't implemented by Apache FOP, yet: table-layout="auto" (on fo:table) (See position 5:2298)
[dbk] juil. 31, 2014 9:32:32 AM org.apache.fop.events.LoggingEventListener processEvent
[dbk] SEVERE: Invalid property value encountered in column-width="proportional-column-width(1)": org.apache.fop.fo.expr.PropertyException: file:/E:/ALM/env/docgen/BUI
[dbk] juil. 31, 2014 9:32:32 AM org.apache.fop.events.LoggingEventListener processEvent
[dbk] SEVERE: Invalid property value encountered in column-width="proportional-column-width(1)": org.apache.fop.fo.expr.PropertyException: file:/E:/ALM/env/docgen/BUI
[dbk] juil. 31, 2014 9:32:32 AM org.apache.fop.events.LoggingEventListener processEvent
[dbk] WARNING: Font "Symbol,normal,700" not found. Substituting with "Symbol,normal,400".
[dbk] juil. 31, 2014 9:32:32 AM org.apache.fop.events.LoggingEventListener processEvent
[dbk] WARNING: Font "ZapfDingbats,normal,700" not found. Substituting with "ZapfDingbats,normal,400".
[dbk] juil. 31, 2014 9:32:32 AM org.apache.fop.hyphenation.Hyphenator getHyphenationTree
[dbk] SEVERE: Couldn't find hyphenation pattern en
[dbk-fo] E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom.fo -> E:\ALM\env\docgen\BUILD\build\source\1\Docgen\docbook.pdf
[dbk] DELETING TEMP FILES... USE property '<property name="ant4docbook.keepTempFiles" value="true"/>' to keep them
[dbk] DELETE FILE : E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom
[dbk] [ERROR] FAIL TO DELETE FILE : E:\ALM\env\docgen\BUILD\build\source\1\Docgen_ant4dbk_docbook.xml.jdom (file not found)
[dbk] process done : E:\ALM\env\docgen\BUILD\build\source\1\Docgen\docbook.pdf

```

# Setting up the Test Level: Deploy to the Web Server

In the previous step, we have created the web archive *docgen.war*. Now, we want to deploy that to the web server which is running IKAN ALM, so that testers may have a look at it. Therefore we need an extra step in our lifecycle, namely a Test Level.

## 5.1. Creating the Test Level

In the Project Administration section, edit the *Docgen* Project.

Go to *Lifecycles > Overview*, edit the BASE Lifecycle and select the *Create Test Level* link at the bottom.

The screenshot shows the 'Create TEST Level' form in the Project Administration interface. The form is titled 'Create TEST Level' and contains the following fields and options:

- Name:** TEST
- Description:** Deploy the generated HTML and PDF as a web archive to the web server.
- Type:** Test
- Locked:** Yes
- Debug:**  Yes  No
- Notification Type:** No notification
- Notification Criteria:** Never
- Requester User Group:** (dropdown menu)
- Pre-Notification User Group:** (dropdown menu)
- Post-Notification User Group:** (dropdown menu)
- Post-Notification Criteria:** (dropdown menu)
- Sequence Level after:** (dropdown menu)
- Life-Cycle:** BASE

At the bottom of the form, there are three buttons: 'Create', 'Reset', and 'Back'.

Creating a Test Level is pretty much the same as creating a Build Level. There are some extra fields for notification which we will neglect for the time being. The new Level will automatically be positioned after the Build level.

## 5.2. Creating the Deploy Environment

Just as for the Build Level, the Test Level is not more than a conceptual step in the Lifecycle. As we require a concrete Machine to deploy our Build result to, we need to link a Deploy Environment to the Level.

Select *Create* on the *Deploy Environments* submenu.

The screenshot shows the 'Create Deploy Environment' dialog box. It has a title bar 'Project Administration > Create Deploy Environment' with a help icon. The main area is titled 'Create Deploy Environment' and contains several fields with red asterisks indicating required fields:

- Name:** TST\_Tomcat
- Level:** TEST (dropdown)
- Machine:** ikan521v (dropdown)
- Build Environment:** build (dropdown)
- Deploy Tool:** ANT1.9.3 (dropdown)
- Deploy Script:** deploy.xml
- Source Location:** C:/ALMDemo/env/docgen/TEST/TST\_Tomcat/source
- Target Location:** C:/ALMDemo/appServer
- Partial Deploy:**  Yes  No
- Debug:**  Yes  No

At the bottom, there are 'Create' and 'Reset' buttons.

This is also very similar to creating a Build Environment.

The deploy will be executed by the IKAN ALM Agent on the selected Machine, and more precisely by the selected (1.9.3) ANT scripting Tool.

We indicate that we deploy the result of our Build Environment by linking it to our Deploy Environment. The defined `deploy.xml` is an Ant script that you can inspect in the sources under `/init_svn/source` and that is also imported in the Subversion trunk of the *Docgen* project.

The script is not very complex: there's a special target for a rollback which we neglect for now; the real action is in the `deploy` target where the web archive `docgen.war` will be copied to an appropriate directory (webapps) under the Tomcat target, and we will use the [Tomcat manager app](#) to “reload” our web archive.

In the source location the Build result previously created will be extracted. The target location must point to our Tomcat web server (TOMCAT\_HOME). That's where the `deploy.xml` script will transfer the build result (`docgen.war`) to.

---

**Note:** If you installed the IKAN ALM Demo, this location will be similar to `ALMDemoHOME/appServer`.

---

Don't bother with the details on this, it just puts the generated HTML and PDF files on the web server so that we can browse to them!

Together with the Deploy Environment, Phases are also linked to it. They will be executed when the Deploy of a Level Request will be executed on the IKAN ALM Agent. You can check those Phases by selecting the

 *Edit Phases* icon next to the Deploy Environment in the *Deploy Environments > Overview*.

### 5.3. Creating the Deploy Parameters for Authentication on the Web Server Manager Application

If you already checked the deploy script, you perhaps noticed that some parameters need to be provided to the reload task applied to the Tomcat manager application: a URL, and a user and password for authentication.

The Tomcat manager application needs to be correctly configured so that authentication by a user with manager-rights is possible. Basically, this comes down to configuring `TOMCAT_HOME/conf/tomcat-users.xml` as follows:

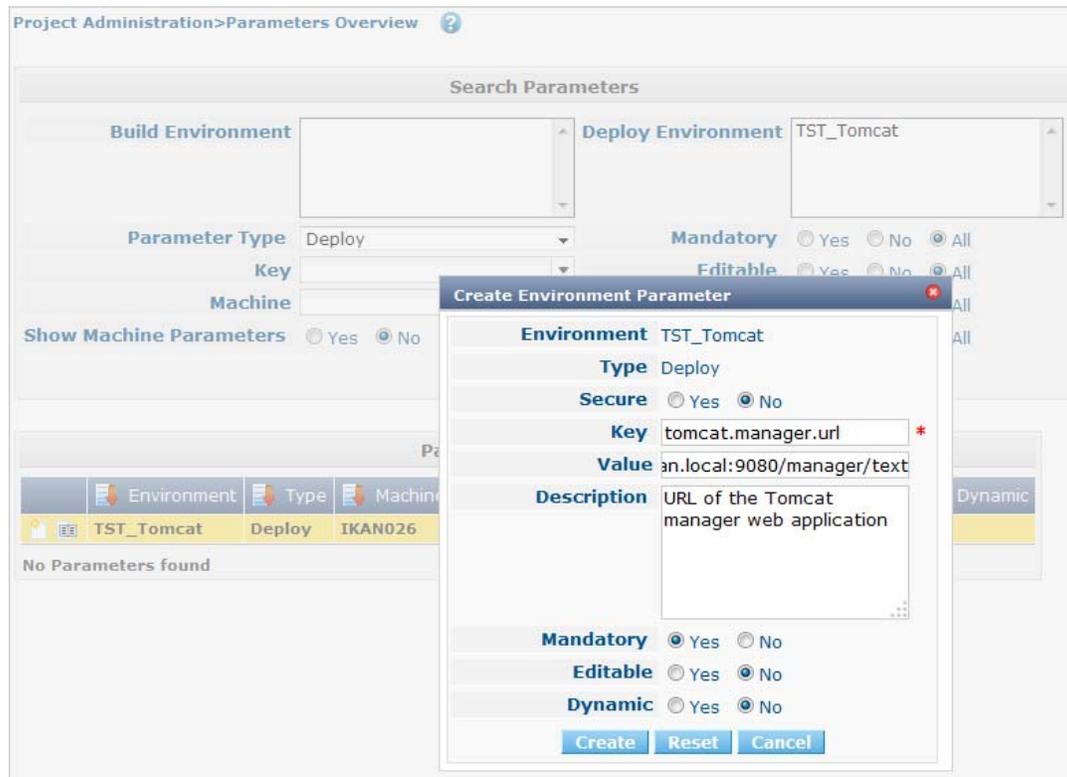
```
<role rolename="manager-script" />
<user name="tomcat" password="tomcat" roles="tomcat,admin,manager-script" />
<user name="role1" password="tomcat" roles="role1,admin,manager-script" />
<user name="both" password="tomcat" roles="tomcat,role1,admin,manager-script" />
/>
```

We need to restart Tomcat to have those settings applied. This can be done by running the shutdown and startup scripts (when Tomcat runs in a prompt/shell) or by stopping and starting the Tomcat service/daemon (when Tomcat runs as a Windows Service/Unix daemon). Check the [Tomcat documentation](#) for more information.

In order to provide the parameters to the script when it is executed, we will define them on our Deploy Environment.

Select *Deploy Environments > Deploy Parameters*.

On the *Parameters Overview*, click the  *Create Parameter* icon in front of the *TST\_Tomcat* Deploy Environment.



We will first create its URL in the `tomcat.manager.url` Parameter as a mandatory parameter. It is using the same machine and port as the URL for the IKAN ALM web application (e.g., `http://localhost:9080`) followed by `/manager/text`.

Next, create the mandatory Parameters `tomcat.manager.username` (value `tomcat`) and `tomcat.manager.password` (value `tomcat`), and make the password a secure parameter (note that you have to enter the password twice).

Parameters Overview										
	Environment	Type	Machine	Actions	Key	Value	Description	Mandatory	Editable	Dynamic
	TST_Tomcat	Deploy	IKAN026		tomcat.manager.url	http://ikan026.ikan.local:9080/manager/text	URL of the Tomcat manager web application			
					tomcat.manager.username	tomcat	User with manager script role			
					tomcat.manager.password	*****	Password			

3 Parameters in 1 Environments found, displaying all

## 5.4. Auditing the Project

Just as for the Build Level, we need to unlock the TST\_Tomcat Level.

Select *Audit Project* from the menu and click the *Unlock* button after you verified the modified Project setup which now contains the Test Level with the TST\_Tomcat Deploy Environment.

## 5.5. Creating the Deliver Level Request

Just like when we created the Build Level Request, we create the Deploy Level Request via the Desktop.

Desktop > Desktop ?  Auto Refresh

1 [icon] +

Desktop Overview							
Desktop Type	Project Stream [Package]	Level	Next Scheduled Request	Latest Level Request	Latest Successful Level Request	Action	Message
Project Stream	Docgen H_1	BUILD		<a href="#">+ H_1_b1[30/01/18 20:33:13]</a>	<a href="#">H_1_b1[30/01/18 20:33:13]</a>		
	Docgen H_1	TEST					

Select the *Deliver* icon in the *Action* Column of the Test Level to create the Deliver Level Request.

Desktop > Create Level Request ?

**Deliver Build** Docgen / H\_I / TEST  
Docgen Project  
Active Build Number: 0

[Back](#) | [Show Additional Info](#)

**Description** Deploy generated HTML and PDF. \*

**Previous Descriptions** [dropdown]

**Requested Date/Time** [calendar]

**Selected Build** 2

Build #	OID	Description	Available on	VCR Tag	End
2	<a href="#">2</a>	Build 1	BUILD	H_I_b2	6/02/18 16:10:54

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

**Parameters**

[Hide Uneditable Parameters](#)

Environment	Key	Value
TST_Tomcat	tomcat.manager.password	*****
	tomcat.manager.url	http://ikan521v.ikan.local:9080/manager/text
	tomcat.manager.username	tomcat

Provide a meaningful description, select the Build that was created earlier on our Build Level, and verify the Tomcat manager Parameters that we created on our Deploy Environment. If everything is OK, click the *Create* button.

## 5.6. Verifying the Deliver Level Request

### Level Request Detail

The Detailed Overview for the Deploy Level Request is similar to the one for the Build Level Request. The differences are to be found in the Phases. If you expand the information panels for the Retrieve Code, Build and Tag Code Phases, you will notice that nothing happened since there is no Build Environment linked to it. Note that the Deploy ended in warning. That is due to the fact that the *Debug* option is still set for the Level.

Desktop>Level Request Detail ?

**Warning** [Docgen / H\\_1 / TEST / Build# 1](#)  
 2: Deploy generated HTML and PDF.  
 Requested by: global on: 7/31/14 10:45:05 AM

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

[← Back](#) | [Refresh](#)

**Actions**

No actions available

**Info**

**Build Number** 1  
**VCR Tag** H\_1\_b1  
**Action** Deliver Build  
**Type** Deploys of archived Build  
**Start** 7/31/14 10:45:05 AM  
**Duration** 00:00:15  
[Show more...](#)

**Builds & Deploys**

	OID	Environment	Machine	Start	Duration
	1	TST_Tomcat	ikan028	7/31/14 10:45:05 AM	00:00:15

**Warning log**

**Phase** Cleanup Work Copy  
**Start** 7/31/14 10:45:21 AM  
**Duration** < 1 sec.  
**Status** **Warning**

**Message**

WorkCopy Location C:/ALM/system/workCopy/2 not cleaned up, because of enabled debugging on the level

[Top](#)

### Deploy Phases log

The different Deploy Phases form the workflow of a Deploy. They are automatically created together with the Deploy environment.

Select the *Phase Logs* tab page. The grey lines on the overview represent the different Phases, the white lines represent the Build or Deploy actions.

Desktop>Phase Logs ?

**Warning** Docgen / H\_1 / TEST / Build# 1  
2: Deploy generated HTML and PDF.  
Requested by: global on: 7/31/14 10:45:05 AM

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

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

**Phase Logs**

> Level Parameters

Phase Name	Start Date/Time	Duration
> + Retrieve Code	7/31/14 10:45:05 AM	< 1 sec.
> + Build	7/31/14 10:45:05 AM	< 1 sec.
> + Tag Code	7/31/14 10:45:05 AM	< 1 sec.
> + Deploy	7/31/14 10:45:05 AM	00:00:15
> + <b>Deploy 1 on machine ikan028</b>	7/31/14 10:45:05 AM	00:00:15
> + Cleanup Work Copy	7/31/14 10:45:21 AM	< 1 sec.

Click the Deploy name (in our example: *Deploy 1 on machine ikan028*) to expand the information panel displaying the Phase details.

▼ + Deploy 1 on machine ikan028 7/31/14 10:45:05 AM 00:00:15

OID 1 Start Date/Time 7/31/14 10:45:05 AM  
Environment TST\_Tomcat Duration 00:00:15  
Machine ikan028 Status **Success**

> Deploy Parameters

> + Transport Build Result	7/31/14 10:45:06 AM	< 1 sec.
> + Decompress Build Result	7/31/14 10:45:07 AM	< 1 sec.
> + Verify Deploy Script	7/31/14 10:45:07 AM	< 1 sec.
> + <b>Execute Script</b>	7/31/14 10:45:07 AM	00:00:13
> + Cleanup Build Result	7/31/14 10:45:20 AM	< 1 sec.

The most important Phase is the *Execute Script* Phase log where we can find the result of the `deploy.xml` script executed by Ant.

Click the *Execute Script* Phase to jump to its *Phase Log*.

**Execute Script** 7/31/14 10:45:07 AM 00:00:13

**Phase Name** Execute Script - 5.5.0 **Duration** 00:00:13  
**Start Date/Time** 7/31/14 10:45:07 AM **Status** Success

> **Message**

▼ **Log**

Download Log

```

startRollBack:

deploy:
[echo]
[echo] For deploying to the webserver, following assumption were made:
[echo] - Tomcat is running
[echo] - The war created on the build level will be copied to C:/ALM/appServer/webapps
[echo] - The manager console of Tomcat can be reached with username="tomcat"
[echo] and password provided by property tomcat.manager.password
[echo] This can be checked in the file: C:/ALM/appServer/conf/tomcat-users.xml
[echo] If username and/or password are not correct, change the values of the
[echo] deployparameters in alm, defined for the deployEnvironment of
[echo] the Test Level: "tomcat.manager.username", "tomcat.manager.password".
[echo]
[copy] Copying 1 file to C:\ALM\appServer\webapps
[reload] OK - Reloaded application at context path /docgen
[echo]
[echo] The demo is deployed successfully. You can verify the demo application by
[echo] using the browser address : http://tomcat_host:tomcat_port/docgen
[echo] Sample: http://localhost:8080/docgen
[echo]

deployActions:

BUILD SUCCESSFUL
Total time: 12 seconds

```

Download Log

Check the final result on [http://tomcat\\_host:tomcatport/docgen](http://tomcat_host:tomcatport/docgen), e.g., <http://localhost:8080/docgen>.

## Docbook article as an IKAN ALM showcase

### Table of Contents

- [Phases](#)
- [Creating a Phase Definition](#)
- [More information](#)

## Phases

When IKAN ALM is running Level Requests, Builds and Deploys, all actions are performed by executing a sequence of Phases. Those Phases are defined in the IKAN ALM database and can be consulted and manipulated in the Phases section of the Global Administration interface. Once they have been defined in Global Administration, Phases may be linked to Levels, Build or Deploy Environments in the Project Administration context.

The IKAN ALM core functionality is performed by so-called "Core" Phases. Those Core Phases can only be viewed, and cannot be altered nor deleted. Consider them an integral part of IKAN ALM.

You can extend this core functionality by creating your own Phases. There are two options:

1. Create a Phase from scratch using the "Create Phase" functionality. In that case, you first specify the name and the version of the Phase, and, next, you choose the script or scripts to be executed by the Phase.
2. Import a Phase that has already been created via the "Import Phase" functionality.

## Creating a Phase Definition

1. Select *Global Administration > Phases > Create*.
2. Fill out the fields in the *Create Phase* panel at the top of the screen. Fields marked with a red asterisk are mandatory:

Also, [http://tomcat\\_host:tomcatport/docgen/docbook.pdf](http://tomcat_host:tomcatport/docgen/docbook.pdf) should open the PDF file we verified earlier in the downloaded Build result.

# CHAPTER 6

---

# Phases

The work on Levels and Environments is done using Phases.

Phases represent specific tasks or actions that must be performed by the system. IKAN ALM comes with a set of “Core” Phases, but you can also create your own Custom Phases, which is even more interesting.

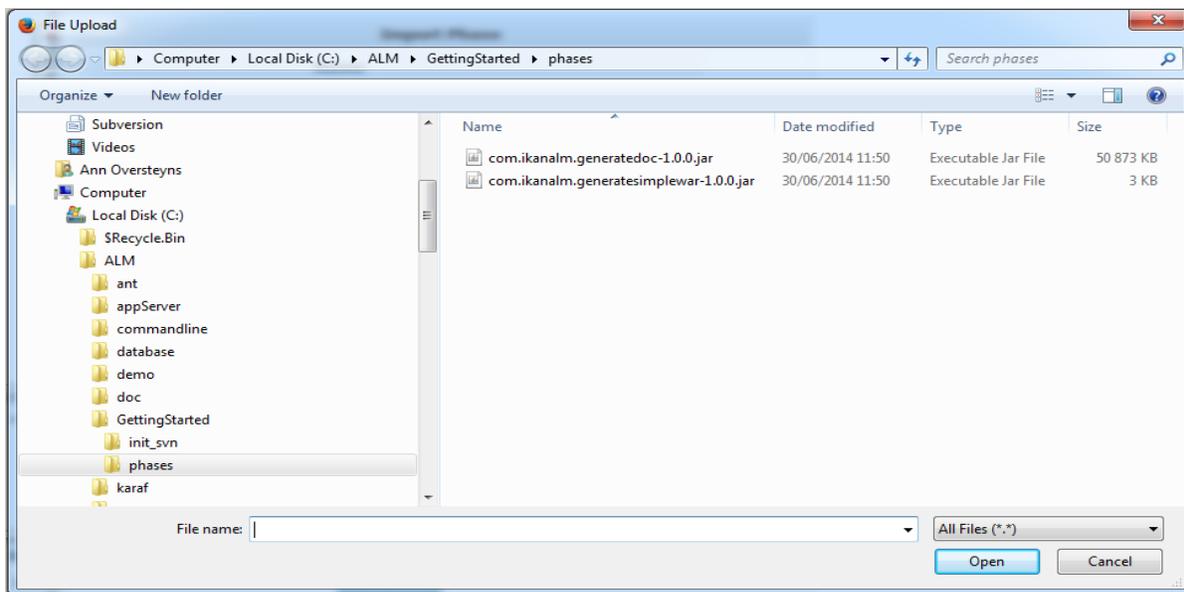
The main advantage of using Phases is that they allow you to customize your project's workflow with reusable building blocks. On top of that, they can be shared and distributed onto local and remote machines.

## 6.1. Global Administration: Importing Phases

For our example, we created two Phases that take over the work done by the build.xml script: *Generate Doc* based on the docbook.xml and *Generate a simple war*. You can find them in the extracted *Getting Started.zip* under /phases.

Let's first import the Phases via *Global Administration > Phases > Import*.

Click the *Select File* button and pick the file associated with the *Generate Doc* phase (com.ikanalm.generate.doc-1.0.0.jar).



Build Environment Yes

Deploy Environment No

Uploaded Files

- generatedoc/generatedoc.xml
- generatedoc/lib/ant4docbook-0.5.0.jar

Select File

Import Overview

Import Phase Parameters					
Name	Default Value	Description	Mandatory	Secure	Integration Type
alm.phase.mainScript	generatedoc/generatedoc.xml		✓		None
alm.phase.extractBundle	true		✓		None
alm.phase.builder			✓		ANT
sourcefile	docbook.xml	Path to Source xml file in docbook format	✓		None
targetfile.name		filename fo the generated document	✓		None
targetfile.type	html	Type (extension) of the generated document. html, pdf, docx, epub and rtf (limited) are supported	✓		None

After the import, you will see all the information concerning the phase, the files contained in it (the generatedoc.xml, which is an Ant script, and the depending ant4docbook library), and the parameters, where sourcefile, targetfile.name and targetfile.type are specific for the *Generate Doc* phase.

Repeat the process for the *Generate Simple War* Phase.

## 6.2. Build Environment: Inserting and Configuring Phases

In the *Docgen Project Administration* section, select *Build Environments > Overview*.

Next, select the  *Edit Phases* icon, next to the “build” Build Environment.

Since the Phases will replace our script, you may remove the *Verify Build Script* and *Execute Script* Phases, by clicking the  *Remove* icon on the *Phases Overview*.

Now we can insert the required Phases. Click the *Insert Phase* link underneath the *Phases Overview*.

The *Generate Doc* Phase will generate the HTML:

Project Administration > Insert Phase ?

Build Environment			
<b>Name</b>	build	<b>Downloadable Build</b>	Yes
<b>Build Suffix</b>		<b>Debug</b>	Yes
<b>Source Location</b>	E:/ALM/env/docgen/BUILD/build/source	<b>Level</b>	BUILD
<b>Target Location</b>	E:/ALM/env/docgen/BUILD/build/target	<b>Machine</b>	ikan028
<b>Build Script</b>	build.xml	<b>Build Tool</b>	ANT1.9.3

Phase to insert	
<b>Phase</b>	Generate Doc - 1.0.0
<b>Fail on Error</b>	<input checked="" type="radio"/> Yes <input type="radio"/> No
<b>Insert at position</b>	2
<b>Next Phase on Error</b>	5. Cleanup Source - 5.
<b>Label</b>	Generate Doc (HTML)
<input type="button" value="Insert"/> <input type="button" value="Cancel"/>	

Phases Overview		Available Phases			
Position	Phase	Phase Name	Phase Version	Execution Type	Author
1	Transport Source - 5.5.0	<input type="radio"/> Generate Simple War	1.0.0	ANT	IKANDem
2	Transport Deploy Script - 5.5.0	<input checked="" type="radio"/> <b>Generate Doc</b>	1.0.0	ANT	IKANDem
3	Compress Build - 5.5.0	<input type="radio"/> Execute Script	5.5.0	CORE	IKAN
4	Archive Result - 5.5.0	<input type="radio"/> Cleanup Result	5.5.0	CORE	IKAN
5	Cleanup Source - 5.5.0	<input type="radio"/> Cleanup Source	5.5.0	CORE	IKAN
6	Cleanup Result - 5.5.0	<input type="radio"/> Archive Result	5.5.0	CORE	IKAN
		<input type="radio"/> Compress Build	5.5.0	CORE	IKAN

Select it from the Available Phases, set Fail on Error to *Yes* and insert it after *Transport Source* Phase (on position 2). When an error happens, we go immediately to the cleanup, by selecting the *Cleanup Source* Phase in the Next Phase on Error. As we will use the Generate Doc Phase twice, first to generate the HTML file and next to generate the PDF file, we can add a label (in our example: *Generate Doc (HTML)*) to distinguish the two.

**Note:** On the *Phases Overview*, the label is displayed as a tooltip when moving the mouse pointer over the eye icon in the last column.

Repeat the steps to insert the Generate Doc Phase once again. Insert it after the previous: this will generate the PDF file. Use the same settings for Fail on Error (Yes) and Next Phase on Error (Cleanup Source) and add a label (for example: *Generate Doc (PDF)*).

Insert the Generate Simple War Phase after the second Generate Doc Phase.

Again, use the same settings for Fail on Error (Yes) and Next Phase on Error (Cleanup Source). As we only use this Phase once, adding a Label is not really necessary.

This will be the result in the *Phases Overview*:

Phases Overview							
			Phase Name	Phase Version	Fail On Error	Next Phase On Error	Description
			Transport Source	5.8.0	Yes	Cleanup Source	
			Generate Doc	1.0.0	Yes	Cleanup Source	
			Generate Doc	1.0.0	Yes	Cleanup Source	Generate Doc (PDF)
			Generate Simple War	1.0.0	Yes	Cleanup Source	
			Verify Build Script	5.8.0	Yes	Cleanup Source	
			Execute Script	5.8.0	Yes	Cleanup Source	
			Transport Deploy Script	5.8.0	Yes	Cleanup Source	
			Compress Build	5.8.0	Yes	Cleanup Source	
			Archive Result	5.8.0	Yes	Cleanup Source	
			Cleanup Source	5.8.0	No	Cleanup Result	
			Cleanup Result	5.8.0	No		

For the Phases to work correctly, we have to adapt some of the specific parameters.

Click the  *View Parameters* icon in front of the Inserted Phases.

First we will adapt the parameters for the *Generate Doc* Phase (the first one for generating the HTML file):

click the  *Edit* link next to `targetfile.name` and set its value to `index`. The `target.type` is defaulted to `html` and must not be changed. With those settings, the Phase will convert the source `docbook.xml` into an `index.html` file.

Phase Parameters						
		Name	Value	Integration Type	Mandatory	Secure
		<code>alm.phase.builder</code>		ANT	✓	
		<code>alm.phase.extractBundle</code>	true	None	✓	
		<code>alm.phase.mainScript</code>	<code>generatedoc/generatedoc.xml</code>	None	✓	
		<code>sourcefile</code>	<code>docbook.xml</code>	None	✓	
		<code>targetfile.name</code>	index	None	✓	
		<code>targetfile.type</code>	html	None	✓	

6 items found, displaying all

Adapt the second *Generate Doc* phase for generating the PDF file: `targetfile.name` = `docbook`, `targetfile.type`=`pdf`, which will result in the generation of a *docbook.pdf* file.

Phase Parameters						
		Name	Value	Integration Type	Mandatory	Secure
		<code>alm.phase.builder</code>		ANT	✓	
		<code>alm.phase.extractBundle</code>	true	None	✓	
		<code>alm.phase.mainScript</code>	<code>generatedoc/generatedoc.xml</code>	None	✓	
		<code>sourcefile</code>	<code>docbook.xml</code>	None	✓	
		<code>targetfile.name</code>	docbook	None	✓	
		<code>targetfile.type</code>	pdf	None	✓	

6 items found, displaying all

Now, adapt the *Generate Simple War* Phase: most of the parameters are fine, just set the appname parameter to *docgen*. With those settings a *docgen.war* will be generated, containing all files in the source *css* directory, and the HTML and PDF file that will be generated by the previous phases.

Phase Parameters					
	Name	Value	Integration Type	Mandatory	Secure
	alm.phase.builder		ANT	✓	
	alm.phase.extractBundle	true	None	✓	
	alm.phase.mainScript	generatewar.xml	None	✓	
	appname	docgen	None	✓	
	includes	css/*.html,*.pdf	None	✓	
	webxml	webapp/WEB-INF/web.xml	None	✓	

6 items found, displaying all

Our Phases are now inserted in the workflow of the Build Environment with the correct Parameter settings. Let's see whether we get the same Build result.

### 6.3. Creating and verifying the Build Level Request

Go to your *Desktop* and create a Build Level Request.

Set the Level Request description to something like: "Test imported Phases".

Verify the Build Level Request by clicking its link on the *Desktop* (see earlier): check the Level Request details, the Build File content and the Build Phases Log.

Build 2 on machine ikan028			7/31/14 11:40:41 AM	00:00:18
OID	2	Start Date/Time	7/31/14 11:40:41 AM	
Environment	build	Duration	00:00:18	
Machine	ikan028	Status	Warning	
Build Parameters				
+	Transport Source		7/31/14 11:40:43 AM	< 1 sec.
+	Generate Doc		7/31/14 11:40:44 AM	00:00:08
+	Generate Doc		7/31/14 11:40:54 AM	00:00:05
+	Generate Simple War		7/31/14 11:40:59 AM	< 1 sec.
+	Transport Deploy Script		7/31/14 11:41:00 AM	< 1 sec.
+	Compress Build		7/31/14 11:41:00 AM	< 1 sec.
+	Archive Result		7/31/14 11:41:00 AM	< 1 sec.
+	Cleanup Source		7/31/14 11:41:00 AM	< 1 sec.
+	Cleanup Result		7/31/14 11:41:00 AM	< 1 sec.

Generate Doc (HTML): click the *Phase Parameters* link to display the used parameters.

**Generate Doc** 7/31/14 11:40:44 AM 00:00:08

**Phase Name** Generate Doc - 1.0.0 **Duration** 00:00:08  
**Start Date/Time** 7/31/14 11:40:44 AM **Status** Success

**Phase Parameters**

Key	Value
alm.phase.extractBundle	true
alm.phase.mainScript	generatedoc/generatedoc.xml
sourcefile	docbook.xml
targetfile.name	index
targetfile.type	html

**Message**

**Log**

[Download Log](#)

```

generate:
[dbk] process done : E:\ALM\env\docgen\BUILD\build\source\2\Docgen_ant4dbk_docbook.xml.jdom
[dbk] use parameter : html.stylesheet = css\bossorg.css
[dbk] COPY C:\Users\ano\.ant4docbook\VO.5.0\css TO E:\ALM\env\docgen\BUILD\build\target\2
[dbk-xslt] Processing E:\ALM\env\docgen\BUILD\build\source\2\Docgen_ant4dbk_docbook.xml.jdom to E:\ALM\env\docgen\BUILD\build\target\2\index.html
[dbk-xslt] Loading stylesheet C:\Users\ano\.ant4docbook\docbook-xsl-1.77.1\xhtml\docbook.xsl
[dbk] DELETING TEMP FILES... USE property '<property name="ant4docbook.keepTempFiles" value="true"/>' to keep them
[dbk] DELETE FILE : E:\ALM\env\docgen\BUILD\build\source\2\Docgen_ant4dbk_docbook.xml.jdom
[dbk] process done : E:\ALM\env\docgen\BUILD\build\target\2\index.html

BUILD SUCCESSFUL
Total time: 7 seconds

```

[Download Log](#)

Generate Simple War:

**Generate Simple War** 7/31/14 11:40:59 AM < 1 sec.

**Phase Name** Generate Simple War - 1.0.0 **Duration** < 1 sec.  
**Start Date/Time** 7/31/14 11:40:59 AM **Status** Success

**Phase Parameters**

Key	Value
alm.phase.extractBundle	true
alm.phase.mainScript	generatewar.xml
appname	docgen
includes	css/,*.html,*.pdf
webxml	webapp/WEB-INF/web.xml

**Message**

**Log**

[Download Log](#)

```

generateWar:
[war] Building war: E:\ALM\env\docgen\BUILD\build\target\2\docgen.war

BUILD SUCCESSFUL
Total time: 0 seconds

```

[Download Log](#)

# Additional Information

In this *Getting Started Guide* you learned how to

- define a Subversion repository,
- create a Project,
- set up the different Build and Test Levels,
- execute requests to build and deploy the Project
- and, last but not least, simplify your workflow by adding and customizing IKAN ALM Phases.

For more in-depth information, refer to the following types of documentation:

- *IKAN ALM User Guide*
- *How to Guide - Using and Developing Custom Phases in IKAN ALM*
- *IKAN ALM Installation Guides*

You can find those documents on our website [www.ikanalm.com](http://www.ikanalm.com).

If you still did not find all the answers to your questions, do not hesitate to contact us at [info@ikanalm.com](mailto:info@ikanalm.com).