



Tableaux[®]

Version: 5.6

© 2011 Incanica Pty Ltd
Tableaux User's Guide

This software and documentation are subject to and made available only pursuant to the terms of the Incanica License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement.

THE SOFTWARE AND DOCUMENTATION ARE PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, Incanica Pty Ltd DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Table of Contents

Part 1	5
Welcome.....	6
Finding Answers.....	11
Before You Start.....	13
Maintenance.....	16
Part 2	18
Basic Elements.....	19
Creating Your First Environment.....	23
Creating Your First Product.....	26
Creating Your First Project-Plugin.....	29
Bringing it all Together.....	33
Part 3	38
Deploying.....	40
Result Codes.....	47
Component Plugins.....	50
Repository Plugins.....	57
Master/Slave Mode.....	61
Release Kits.....	65
Tokens.....	72
Part 4	78
Goals.....	79
Project Schedule.....	83
Reporting Graphs.....	86
Reporting.....	88
Searching.....	90
Part 5	93

User & Group Management.....	95
Security Model.....	101
Object Attributes & Permissions.....	104
Keywords.....	107
System Configuration.....	110
Audit Logs.....	114
Digital Signatures.....	116
Deleted Object Repository (Graveyard).....	118
Licensed User Report.....	120
Part 6.....	122
Approvals.....	123
Scheduler.....	126
Command-Line.....	129
Troubleshooting.....	132
Glossary.....	134
Index.....	137

Part 1

NUTS & BOLTS

Welcome.....	6
Contents of Package.....	7
System Requirements.....	7
Installing Tableaux.....	7
Licensing Tableaux.....	8
Testing the Installation.....	8
Optional Extra - PostgreSQL.....	9
Optional Extra - MySQL.....	9
Finding Answers.....	11
About This Guide.....	12
Basic Terms.....	12
Online Tutorial.....	12
Online Help.....	12
Technical Support.....	12
Before You Start.....	13
Settings.....	14
Connecting to your LDAP Server.....	14
Setting Directories.....	14
Installing Repository Plugins.....	15
Maintenance.....	16
Database Maintenance.....	17
Backups.....	17

Welcome

Tableaux is a tool designed to help you become more effective at developing software. It becomes the hub of your software development process to provide a consistent view of the state of development. It helps you overcome the often conflicting goals of:

- Reducing the software development time schedule
- Increasing the quality of deliverables by making fewer mistakes

Its secondary goal is to break down barriers between groups involved in the software development process and improve the workflow between the groups.

You should familiarise yourself with the contents of this manual. In particular, the Quick Start Primer will get you started

Contents of Package

Tableaux comes with everything you need to get up and running. This includes:

- Installation CD
- Tableau's User's Guide

System Requirements

For an up to date list of requirements for the product please refer to the website at <http://www.incanica.com/tableaux.html>

In general you will require a system that has the following minimum specifications:

Operating System:

- Unix: Linux, Solaris, FreeBSD, AIX, etc
- Microsoft Windows: NT, XP, 2000, 2003, Vista, 7, etc

Hardware:

- 500MHz CPUs or above, 1GB RAM
- 1GB available disk space

Software:

- Java JDK (not the JRE) 1.6+

Installing Tableau

There are 3 main components that make up the Tableau system. They are:

- The application server (Tomcat)
- The database (McKoi or PostgreSQL)
- The deployment scripts

You must have available either the Unix or Windows version of Tableau. The Unix version comes packaged as a .tar.gz file. The Windows file comes packaged as a .zip file.

Unix installation

Before beginning you should create a user for the owner of Tableau. Normally this user would be "tableaux".

You must also have the Java 1.6+ executable in your path and the JAVA_HOME variable set.

Unpack the .tar.gz file into a temporary location:

```
# mkdir /tmp/tmptableaux
# cd /tmp/tmptableaux
# gunzip -c <source file.tar.gz> | \
  tar -xf -
```

Run the installation command "install.sh" as the root user. This script will prompt you for the locations of various components. In most cases you may simply accept the defaults:

```
# ./install.sh
Please enter the location to install
Tableau into: [/usr/local/tableaux]:
```

```
Please enter the unix owner of
Tableau: [tableaux]: tableau
```

```
#####
Installing the database component
...
```

```
Installed the tableau files.
```

The installation process will have attempted to create a start/stop script in `/etc/init.d` on your system. If your system does not have this directory then the file will not be created.

If required, a template start/stop script is located in the `.tar.gz` file called `tableaux.sh`. Customise as required.

Windows Installation

Create a directory for the location at which you will install Tableau. We recommend the following directory:

```
c:\Program Files\Tableaux
```

Unpack the `Tableaux.zip` file into the directory. Execute the following script:

```
c:\Program Files\Tableaux\install.bat
```

This will install two services, one to automatically start the database and one to automatically start the application server.

Java Home

Tableau is a Java application. You need an environment variable called `JAVA_HOME` set before Tableau will run. You can set `JAVA_HOME` in a few places, including:

- In the `tableaux/tomcat/bin/catalina.sh` startup script.

- On Unix, in the "tableaux" user's profile (`~tableaux/.profile`)
- On Windows, in the "tableaux" user's environment.

Licensing Tableau

Tableau requires a license key to run. Please contact Customer Support using the details from our web site <http://www.incanica.com> to obtain a key.

You will be given a file containing your key. Place this file into the directory you entered during the Tableau installation process.

Please contact Customer Support if you need to change the location of the license key after installation.

Testing the Installation

Start Tableau by either enabling the services on Windows or running the `/etc/init.d/tableaux` script on Unix.

Point your browser to:
`http://<servername>:8080/tableaux`

You should see a login screen. If the login screen does not appear for you then please refer to the troubleshooting section later in this guide.

Once you see the login screen then you may log in as the default administrator user. The details of this user are:

- Username: admin
- Password: admin

For security reasons you should change this password as soon as possible.

Optional Extra - PostgreSQL

Tableaux comes bundled with a database called McKoi. This database has size limitations. If you will be running a large site (scores of projects) then you might consider running PostgreSQL in McKoi's place.

The website:

<http://www.postgresql.org/download/> contains the latest version of PostgreSQL for you to download.

The website:

<http://www.postgresql.org/docs/manuals/> contains installation instructions for your platform.

You may install the database either on the Tableaux server or on an alternative one. Once installed, make the following changes:

- Inside `tableaux.tar.gz` is a file called `db_scripts.zip`. Inside that file is a script called `create_pg_db.sh`. You may have to customise this script to your installation before running it.

For Unix installations:

- Obtain the JDBC driver from <http://www.postgresql.org/download/> and copy it to `<install dir>/tomcat/common/lib`.

- Modify `<install dir>/tomcat/webapps/tableaux/WEB-INF/web.xml` and change the driver, host and url sections.

For Windows installations:

- Obtain the JDBC driver from <http://www.postgresql.org/download/> and copy it to `<install dir>\tomcat\common\lib`.
- Modify `<install dir>\tomcat\webapps\tableaux\WEB-INF\web.xml` and change the driver, host and url sections.

Finally, restart Tableaux.

Optional Extra - MySQL

You may also optionally run MySQL in place of PostgreSQL or McKoi.

The website: <http://www.mysql.com/downloads/> contains the latest version of MySQL for you to download.

The website: <http://dev.mysql.com/doc/> contains installation instructions and reference manuals for your platform.

Alternatively, your OS platform may already have MySQL installed, or provide a packaging system to easily install it.

You may install the database either on the Tableaux server or on an alternative one. Once installed, make the following changes:

On the new MySQL server:

- Copy all `db/mysql/*.sql` files to the MySQL server.
- In `mysql_create.sql`, modify the password as required.
- To generate the database, run:
`/usr/local/mysql/bin/mysql -u [admin user] < ./mysql_create.sql`

On the Tableaux server:

- Fetch JDBC driver from:
<http://dev.mysql.com/downloads/connector/j/>
- Copy the `jdbc.jar` file to:
`tomcat/common/lib`
- Modify the `tableaux.properties` file with the following details:

Driver: `com.mysql.jdbc.Driver`

URL: `jdbc:mysql://localhost/tableaux`

Password: as set in `mysql_create.sql`

Finally, restart Tableaux.

Finding Answers

Tableaux is an extremely flexible tool to help with the management of your software development process. You should familiarise yourself with the basic controls that Tableaux offers in order to extract the best from the tool.

Incanica provides a number of sources of information on the Tableaux tool. This user guide is the best source of information to answer most questions you may have. The online help system may also be of some use.

If your questions are not answered by this guide or the online help then this chapter discusses methods to contact Incanica for technical support.

About This Guide

This guide is designed to provide all the information you need to know to fully operate Tableaux in an enterprise environment. The guide provides a comprehensive index at the back to make it easy to find your desired topic.

Please note that this guide only provides general information on how to integrate and automate your existing and future tools into the system. It does not cover the operation or integration of any specific tool set.

Basic Terms

The following terms are used extensively throughout this user guide.

Click - A single press and release of the mouse button.

Back - Most screens present a specific "Back" button near to the top of the screen. Otherwise, this means the browser's history back button.

Online Tutorial

New users to Tableaux may not always be given adequate training before engaging the tool. Every user is presented with an online tutorial after the first log-in into the system.

This tutorial can be accessed again by selecting "**Help -> Tutorial**" from the menu at the top of the Tableaux screen.

Online Help

Tableaux provides extensive online help. Once you are logged into the system, at the top of the screen you can select "**Help -> Help**".

Technical Support

Before contacting Incanica Technical Support, please ensure that your question is not already answered by this guide or by any online help.

The contact information for Technical Support can be located online at: <http://www.incanica.com>

Before You Start

There are several items that you need to customise in Tableaux before you can seriously start using the tool.

These items are the basic configuration required to ensure that Tableaux is talking to the right systems and tools.

Settings

Tableaux provides many options that change its behaviour. These settings are available for any Tableau administrator via the "**Administration -> System Configuration**" menu.

The most important settings initially are detailed below.

Connecting to your LDAP Server

Many organisations utilise a directory server to manage authentication and authorisation for their employees. Almost all directory servers can communicate via the LDAP protocol, so Tableau was built with this facility in mind.

To have Tableau authenticate to a directory server that uses LDAP, choose "**Administration -> System Configuration**" from the menu, then the Authentication tab.

Enter the following details:

- The host of the directory server
- (Optionally) the port (defaults to 389)
- The base location of the user objects in the LDAP tree
- The attribute in the user objects to search for (commonly "uid" for iPlanet/Sun Directory server, or "sAMAccountName" for Active Directory).
- (Optionally) a username and password for Tableau to connect to the directory server to search for the specified user object, if no anonymous lookups are allowed.

- Use LDAP connection pooling. This speeds up LDAP authentication and authorisation requests, at the expense of keeping several constant connections open to Directory Server.
- Auto-create user profiles. If checked, Tableau will automatically create a user account under specific conditions (see Chapter 22 - Creating a User).

LDAP authentication is only used when a user's password is blank. Setting a password for a user is an "over-ride" to their LDAP password - this is useful for Tableau administration accounts.

To test that the LDAP authentication is working, hit the "Test" button. Ensure you save the details first. Alternatively, create a user (see Chapter 22 - Creating a User) and leave the password blank. Attempt to log in as the user.

Setting Directories

Tableau will provide default values for several directories installed with the tool. If you wish to move these components then you must also change their location in Tableau.

From the menu, choose "**Administration -> System Configuration**", then the Directories tab. You may customise:

- The location that Tableau stores text files containing deployment history
- The location that Tableau looks for deployment scripts

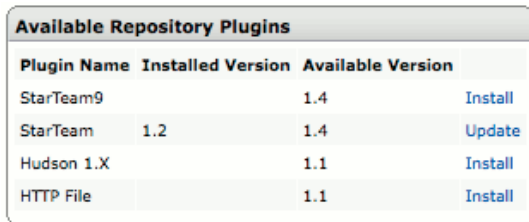
- The location that Tableau creates a temporary “sandpit” during deployment (usually /tmp or c:\temp)

The purpose of these directories is explained later, suffice it to say that the history directory should have at least 1GB free on its filesystem.

Installing Repository Plugins

For Tableau to interface with Version Control Systems (also known as Software Configuration Management tools, or SCM tools) it requires plugins to be installed.

From the menu, choose “**Administration** -> **Repositories**”. Clicking on the “Check For Updates” button will force Tableau to check for any new available plugins for your version of Tableau.



Plugin Name	Installed Version	Available Version	
StarTeam9		1.4	Install
StarTeam	1.2	1.4	Update
Hudson 1.X		1.1	Install
HTTP File		1.1	Install

Choose a plugin that matches the SCM(s) you are running in your organisation and either Install or Update it.

After installation, click on the name of the plugin to configure its parameters. These parameters will depend entirely on the plugin you've chosen and the configuration of your infrastructure.

Maintenance

Tableaux is relatively maintenance free and should normally not require much on-going attention. However there are still some areas of Tableaux that you will need to address, specifically in terms of the underlying database.

Database Maintenance

Tableaux in general does not delete many items from its back-end database. However, like most systems, a regular vacuum of the tables will keep the database in the best possible health.

The method of doing this depends on the database you are running.

If you are running the built-in McKoi database

There is a script provided that looks after the health of the database. On Unix you will find it here: `<install dir>/db/compact_tables.sql` and on Windows: `<install dir>\db\compact_tables.sql`

You could create a scheduled job that invokes the script via the following command line:

```
cd <install dir>/db; ./load.sh \
compact_tables.sql
```

If you are running a PostgreSQL database

The backup script described below covers the health of the database.

Backups

Even the best computer systems suffer from failures, so backups are extremely important for protecting the long-term health of your Tableau system.

There are no special files in the Tableau software, so your regular backup software will suffice to archive its files.

You may simply backup the entire directory Tableau is installed to. By default this is

```
/usr/local/tableaux
```

or

```
c:\Program Files\Tableaux
```

When backing up the database, you should either halt the database, or otherwise perform a database dump and back it up.

On the McKoi database, use the following to dump the database to a file:

```
cd /usr/local/tableaux/db
./sql
    CALL SYSTEM_MAKE_BACKUP('/path/1');
```

If you are using PostgreSQL then run the following script:

```
/usr/local/tableaux/db/
postgres_backup.sh
```

There is also an associated restore command:

```
/usr/local/tableaux/db/
postgres_restore.sh
```

Part 2

Quick Start Primer

Basic Elements.....	20
Menu Bar.....	21
Home Page/Portal.....	21
Products.....	21
Goals.....	22
Environments.....	22
Work Screen.....	23
Work Screen Views.....	23
Creating Your First Environment.....	25
Environment Pools.....	26
Environment.....	26
Polishing the Environment Pool.....	26
Creating Your First Product.....	28
Product.....	29
Component.....	29
Creating Your First Project-Plugin.....	31
Creating a Plugin.....	32
Actions.....	32
Parameters.....	33
Bringing it all Together.....	35
Finishing the Component.....	36
Create a Scenario.....	36
Modify the Parameters.....	37
Choosing a Repository.....	38
Deploying the Component.....	38

Basic Elements

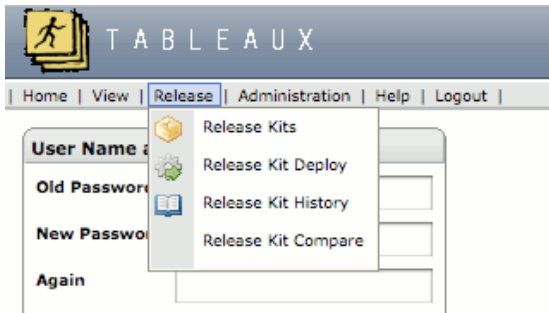
This chapter aims to familiarise you with the basic entities and controls in Tableaux. These elements form the bulk of the Tableaux system and an understanding of them is mandatory for all users of Tableaux.

These basic elements include key screens in the application, the menu bar which is present on every screen, and several of the main objects in the system.

Menu Bar

Tableaux is a web-based system. However it attempts to emulate a desktop application by providing a menu bar.

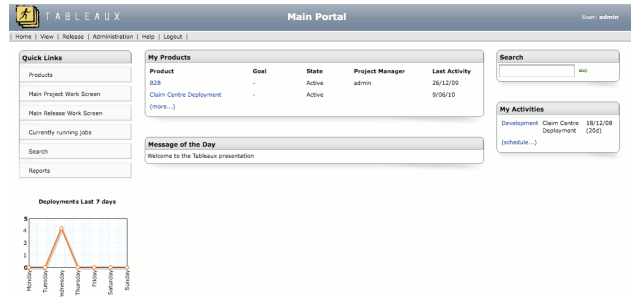
The menu bar is always available from every screen. The items in the menu bar depend entirely on your access privileges. The items may also change slightly, depending on the screen you are currently in.



Every function in Tableaux is accessible via the menu bar.

Home Page/Portal

Each time you log into the system, you are taken to the home page. The home page is a portal which presents you with quick access to many of the main features in Tableaux.



The portal presents you with the following items:

- Some quick links to various major functions in Tableaux
- A list of the products you are currently working on
- A search box
- A graph showing overall activity for the last seven days
- The message of the day
- The activities you are currently assigned to.

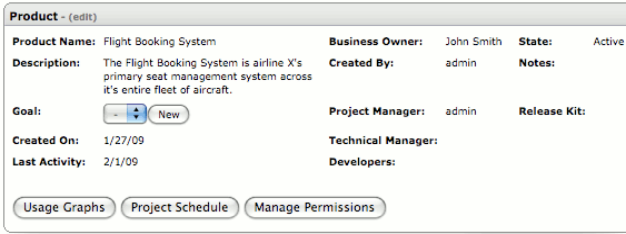
Products

Tableaux is a multi-user/multi-project system. It provides efficiencies of scale as each project, and the number of projects increases. To organise components, and activities on those components, we introduce the concept of a Product.

A Product is a basic container that lets you collect and manage related components. Components are physical entities that represent the digital artefacts and actions that combined comprise your project.

A Product contains:

- General information, such as the product name, creation date, etc
- One or more Goals (see below)
- A component manifest describing the components that make up the product
- A list of users who are working on the Product



Goals

A Product, throughout its lifetime, may undergo several (or continuous) revisions. Tableaux makes it possible to account for the work effort put into each stage or revision separately. The mechanism that allows this in Tableaux is the "Goal".

Goals allow:

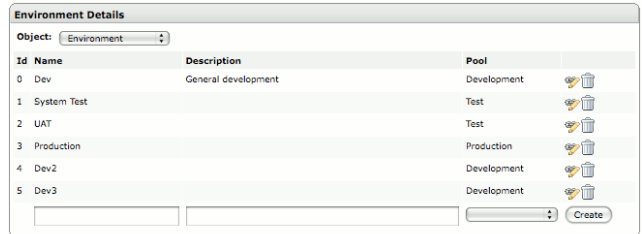
- Concurrent development of a Product by two or more teams without interfering with each other
- A means to contain the work required to reach a particular version of the Product
- A way of describing the various revisions that a Product has undergone (and which revision(s) are current).

Environments

Tableaux is a software development life-cycle management tool. It helps you manage the complexity of moving your product from design to maintenance. Your product during that time will move through various stages of development and testing.

Tableaux calls the various stages that a product moves through "Environments". Examples of Environments are "Integration Testing", "System Testing" and "Production".

Environments usually relate to physical servers, a cluster of servers, or even virtual or shared servers. However, it is possible to create Environments simply as placeholders or way-points in your workflow.



Tableaux provides a framework for the interaction of Products (and their components) with Environments.



Environments are organised by "Environment Pools". Usually you would create the following Pools:

- Development
- Testing
- Production

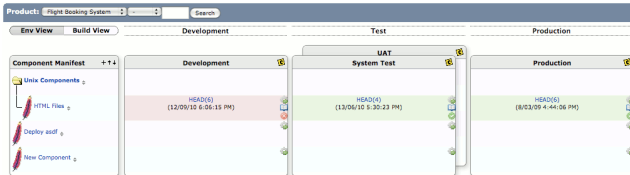
These are just suggestions, and you are free to create as many pools as you like.

Work Screen

The main work screen presents a matrix that shows the Components of each Product and the versions that are in each Environment.

The main work screen is accessible in two locations:

- The work screen is available in its entirety from the menu bar



- Each Product also has a customised view of the work screen (picture on previous page).

They are essentially the same screen and changes made in one are reflected in the other.

There are a few navigation notes for the Work Screen:

- When you have multiple Environments in the Pool, they become stacked behind each other. Click anywhere on a hidden Environment to bring it to the front.
- The Component Manifest contains a folder hierarchy. Click on the folder icons to open them up.
- You can right-click on each Component and folder to get a context sensitive-menu.

Work Screen Views

There are two different views available on the work screen.

Environment View

The default view into the work screen is the "Environment View". This view displays the environment matrix, clearly showing the state of each environment.

Build View

The alternative view in the work screen is the "Build View". Click on the "Build View" button to switch over to this view.













Build	Tag	Environment	Start Date	Time	User	Result	Cost	Keywords
1	HEAD01	Development	12/09/10 8:00:13 PM	14	admin	Success		
2	HEAD01	Development	20/10/10 1:27:46 PM	24	admin	Success		
3	HEAD01	Production	8/03/09 4:44:04 PM	14	admin	Success		MC 1234
4	HEAD01	Development	8/03/09 1:56:21 PM	24	admin	Success		
5	HEAD01	Development	8/03/09 1:52:39 PM	24	admin	Success		
6	HEAD02	System Test	18/06/10 9:30:13 PM	16	admin	Success		APPROXISE ADMIN
7	HEAD02	Development	8/03/09 1:51:36 PM	24	admin	Success		

The Build View displays the deployments from a component/build centric point of view. Click on any component to view its builds.

Creating Your First Environment

This chapter will run you through the creation of your first Environment. Remember that an environment is a virtual entity that represents a stage in the software development process. Most often an Environment in Tableau is representative of one or more physical servers. It's also possible they are simply used as a placeholder.

Environment Details
Object: Environment

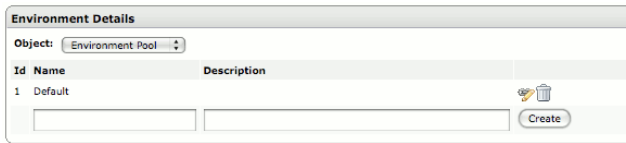
Id	Name	Description	Pool	
0	Dev	General development	Development	 
1	System Test		Test	 
2	UAT		Test	 
3	Production		Production	 
4	Dev2		Development	 
5	Dev3		Development	 

Environment Pools

Tableaux makes managing your development pipeline a breeze. With a proper environment management system in place, the number of environments in your pipeline may grow considerably.

With this in mind, you need to organise your environments in such a way as to allow for proper Configuration Management checks and balances. How Tableaux does this is through Environment Pools.

From the menu bar, choose "**Administration -> Environment Pools**".



Tableaux creates a single pool for you called "Default". A pool is only visible to users if there are environments in it, so the Default pool will remain hidden unless you use it.

We will not use the Default pool. Instead we will create three new pools. In the Name field enter "Development". Enter some text for the description field and click on Create. Your first pool should appear in the list, like this:

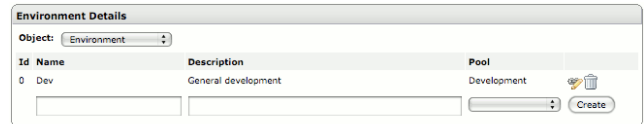


Similarly, create pools called "Test" and "Production".

Environment

The same screen as we used above is also used to manage environments.

In the Name field enter "Dev". Enter some text for the description field. Choose "Development" for the pool. Click on Create. Your first environment should appear in the list, like this:



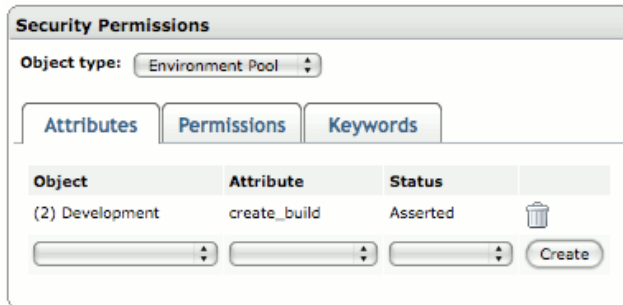
Similarly, create the environments "System Test" and "UAT" in the test pool. Create "Production" in the production pool.

Polishing the Environment Pool

There's one last action to take for the environment pool. We need to define one of the pools as a receptacle for new builds. Normally Tableaux does not allow you to deploy a build into an environment unless it has previously been deployed into a lower environment. However, we need to define at least one environment that you can introduce new builds into.

Select "**Administration -> Permissions**" from the menu. This brings up a screen where you can modify the attributes and permissions for objects within Tableau (see Chapter 24 - Object Attributes & Permissions).

In the subsequent screen, select "Environment Pool" from the Object type drop-down. Then set the next drop-downs as picture below, then click on the Create button:



Chapter 7

Creating Your First Product

This chapter will run you through the creation of your first Product. After this chapter you will have a placeholder for you to start your actual project.

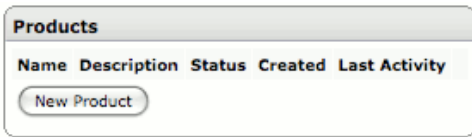
The screenshot displays the Tableau interface for a product named "Flight Booking System". The top section shows product details in a table format:

Product - (edit)		
Product Name: Flight Booking System	Business Owner: John Smith	State: Active
Description: The Flight Booking System is airline X's primary seat management system across it's entire fleet of aircraft.	Created By: admin	Notes:
Goal: [Dropdown] [New]	Project Manager: admin	Release Kit:
Created On: 27/01/09	Technical Manager:	
Last Activity: 6/07/11	Developers:	

Below the details, there are three environment views: "Development", "Test", and "Production". Each view shows a "HEAD" commit with a date and time. The "Development" view shows HEAD(6) on 12/09/10 at 6:06:15 PM. The "Test" view shows HEAD(4) on 13/06/10 at 5:30:23 PM. The "Production" view shows HEAD(6) on 8/03/09 at 4:44:06 PM. On the left, there is a "Component Manifest" panel with a tree view showing "Unix Components", "HTML Files", "Deploy asdf", and "New Component".

Product

From the menu select "View->Products". This will take you to the product listing screen.

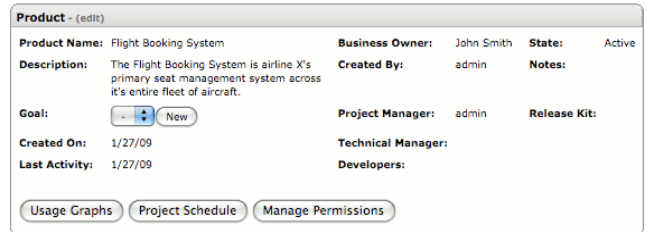


Click on the "New Product" button. If this button does not appear for you then you must log on as a user with administrator privileges (see Chapter 22 - User & Group Management).

Enter the following details for your product:

- Product name
- Description
- Notes

Click on the Save button. You will be taken to the main Product view screen.

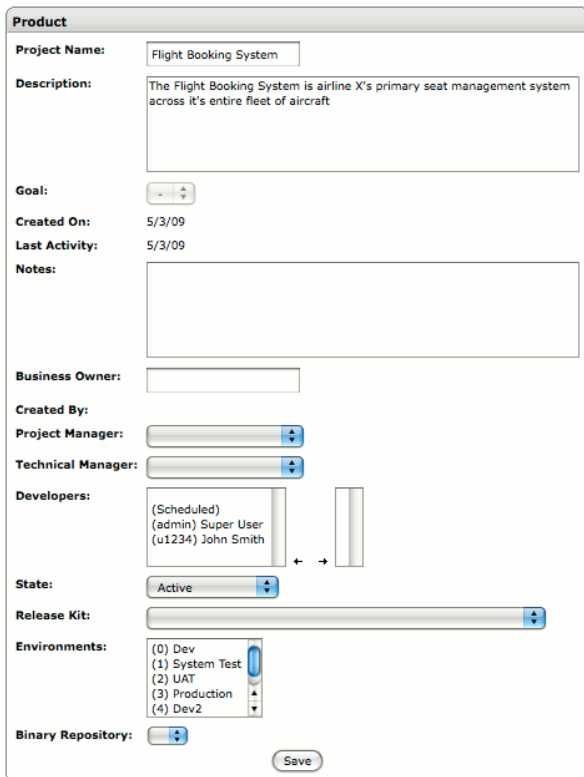


You may notice that a default Goal called "-" is created for you. Every Product has this Goal and for many projects this will suffice.

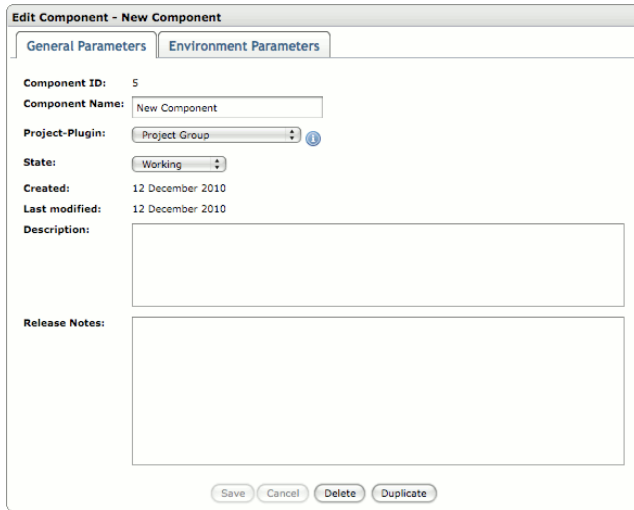
Your Product is now ready for use.

Component

From the main Product screen, there is a box titled "Component Manifest". Click on the + icon in this box. This creates your first component.



This component is a container. It is represented by a closed folder icon. Every new component has the name "New Component". We will now change the name of this component. Click on the name of the new component. This will bring you to the component edit screen.



This is a complicated screen, but for now we'll only deal with the first tab. Change the text "New Component" to "Unix Components" and click on the Save button.

You have now changed the name of the component. You will be back at the Product screen which now displays the new component name.



At this stage, if you click on the folder icon, nothing will happen, because this folder contains no sub-components. So we'll create one now.

Click on the + icon again to create another component. Change the name of the new component to "HTML Files". Next we will move the new component into the "Unix Components" container. Next to the component name is a little icon with up/down arrows (⇅). Click on this icon and the cell becomes grey. Move your mouse anywhere inside the "Unix Components" cell. You will notice that the mouse arrow turns into a crosshair.



Click the mouse. You will be prompted whether you would like to move the component inside, or below the folder. Click on "OK" to move it inside.

It will now appear as though "HTML Files" has disappeared. However it has moved inside the "Unix Components" container. If you click the folder icon it will open it up and expose the "HTML Files" component.

This is the method by which you create and organise components in a Product.

Later on, we'll show you how to make the components perform an action in an environment.

Creating Your First Project-Plugin

Earlier on we created a component for a product. At that stage, the components did not perform any actions.

With the environments defined, we can now have the components perform an action in that environment.

The type of actions that a component can perform in an environment depends entirely on the plugin that it uses.

Creating a Plugin

From the menu bar, choose "**Administration -> Project-Plugins**". On the resulting screen a few plugins have been pre-defined for you.

Click on the "New" button.

On the following screen you must enter the bare essentials of the Plugin. Enter "Deploy HTML" for the name, enter anything in the Description field, and finally choose "standard.gif" for the image. Next, click on the Actions tab.

Actions

Each Plugin is defined by two entities. They are Actions and Parameters.

Actions associate a custom system command to run with the Plugin. When an Action is nominated to run during a deployment, Tableau invokes the system command associated with it; the Action becomes a symbolic placeholder for the system command.

In the Plugin edit screen, the second section allows you to define the Actions. Enter "Deploy" for the name, and "deploy_files" for the script name. Click on "Create".

You'll notice that your Action cannot be created because the system command cannot be located.

In Chapter 3 - Before You Start of this user guide you pre-defined the location of these commands. By default, the location of the commands on Unix is:

```
/usr/local/tableaux/scripts
```

On Windows the location is:

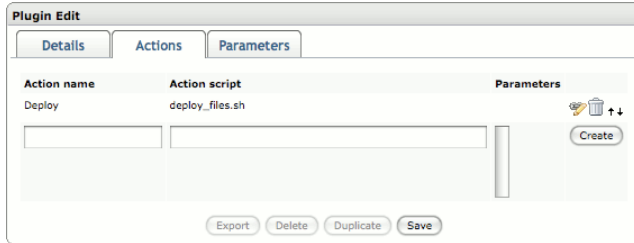
```
c:\Program Files\Tableaux\scripts
```

Create a file in the directory called "deploy_files.sh" or "deploy_files.bat" depending on your platform. For either platform, include the following text in the file:

```
echo Hello World Plugin Action
```

Ensure that the file is executable.

You can now revisit the Action screen and re-enter the details to create the Action. This time the action will create correctly.

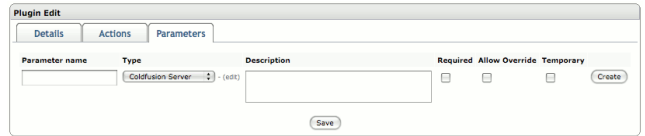


The system command is invoked with specific parameters which determine how the command performs its job in the nominated environment. In this case we've created a Hello World action that does not use parameters to do its job. In Chapter 12 - Component Plugins we'll cover the topic of creating useful actions.

Parameters

Parameters define the various input parameters that are required by the system commands associated with each action.

Each Component that uses a Plugin must provide values for the parameters for each environment it will deploy to. Back to Chapter 7 - Creating Your First Product we introduced the Component Edit screen. This screen allows you to modify these values (we'll enter values for those in the next chapter). Click on the Parameters tab.



Whilst in the Plugin edit screen, you'll notice that each parameter has the following fields:

- Name - The name of the plugin appears on the Component Edit screen as a description of the field.
- Type - This defines the field type that is presented for you to enter a value into. Examples are a text field, a password field, a checkbox, etc.
- Description - This can be any text.
- Required - Determines if the system command associated with the Action requires this parameter to perform its job, or whether it is merely optional.
- Allow Override - Determines if the user can override the value of this parameter at deployment time. Usually the values are pre-determined at configuration time.
- Temporary - Values are never stored in the history. Use this checkbox to define fields to be used for one-time, secure values such as sensitive passwords.

Getting back to our example Plugin called "Deploy HTML" we could imagine that our action would have to know:

- The host to deploy to
- The files to deploy

In the Plugin edit screen, create two parameters like this:

Chapter 8 Creating Your First Project-Plugin



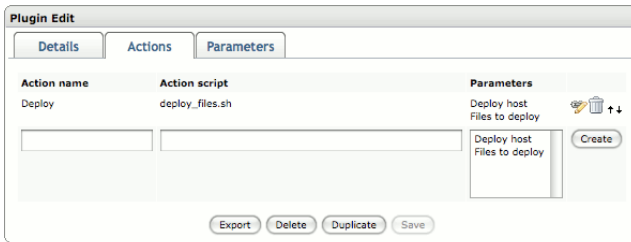
The screenshot shows the 'Plugin Edit' dialog with the 'Parameters' tab selected. It contains a table of parameters and a 'Save' button at the bottom.

Parameter name	Type	Description	Required	Allow Override	Temporary
Deploy host	text	The destination web server, either by name or ip address	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Files to deploy	text	The files to deploy to the web server	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table, there are input fields for 'Deploy host' and 'Files to deploy'. The 'Files to deploy' field has a dropdown menu showing 'Coldfusion Server' and a 'Create' button. A 'Save' button is located at the bottom center.

Finally, you need to associate the two parameters with the Action we created earlier.

Re-enter the Actions tab and edit the action. Add the two parameters to the action from the select box.



The screenshot shows the 'Plugin Edit' dialog with the 'Actions' tab selected. It displays the action details and a list of parameters to be associated with the action.

Action name	Action script	Parameters
Deploy	deploy_files.sh	Deploy host Files to deploy

Below the table, there are input fields for 'Action name' and 'Action script'. The 'Parameters' column has a list of parameters: 'Deploy host' and 'Files to deploy'. A 'Create' button is located to the right of the parameters list. At the bottom, there are buttons for 'Export', 'Delete', 'Duplicate', and 'Save'.

Click on the Save button.

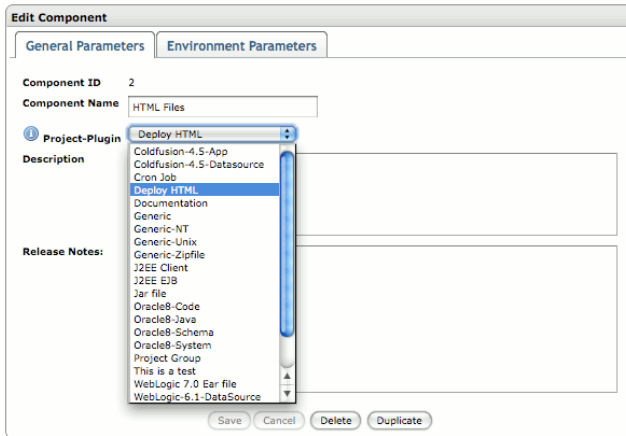
Bringing it all Together

We've now introduced all the concepts needed to get Tableau to do something. Tableau is considerably more feature-packed than this, but these are the basic elements that you need to know.

This chapter will now tie everything together and make you familiar with the end-to-end process.

Finishing the Component

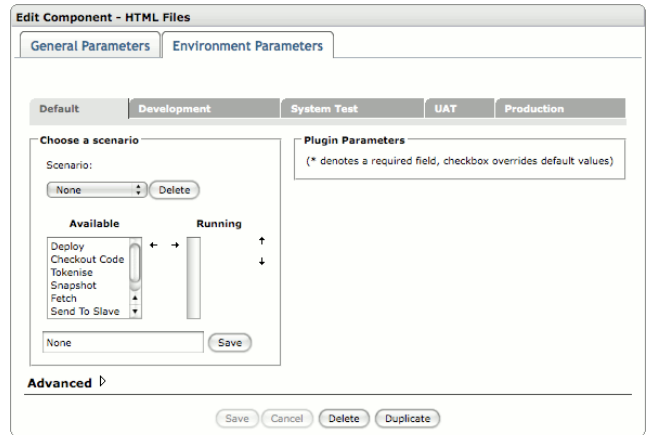
In the product screen you'll want to edit the "HTML Files" Component once again. This time, however, you'll be changing the Plugin used for the Component. There is a drop-down list for the Plugin. This list now includes the Plugin we created in the last chapter.



Change the drop-down to the new Plugin and click on Save. The first thing you will notice is that the icon for the Component in the work screen has changed. The icon adjacent a component always reflects the Plugin used by the component.

You'll want to, once again, edit the "HTML Files" Component. Click on its name to open the edit screen then click on the Environment Parameters tab.

One thing to notice about the following screen is that despite us changing the Plugin to Deploy HTML there are still no plugin parameters to fill in.



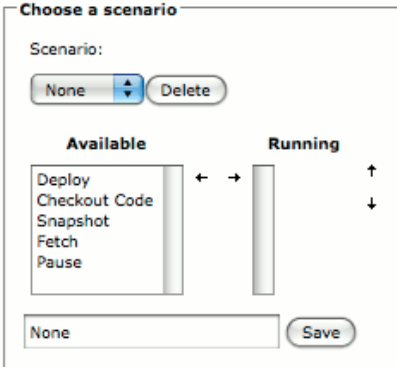
Create a Scenario

In a previous chapter, we defined the Actions inside of the Deploy HTML plugin. Our example only includes a single Action, however it is more common for a Plugin to include multiple Actions that can be run in varying combinations. For example, a common scenario is to run the following actions:

- Check out code from the appropriate repository
- Compile the code using the appropriate compiler
- Deploy the compiled binaries to the destination
- Unit-test the newly deployed binaries.

These combinations are such a common occurrence that TableauX allows you to store these usage patterns as Scenarios.

The left-hand side of the edit screen is dedicated to Scenario creation. Initially there will be no Scenarios available (as indicated by the None in the drop-down box).



The premise of creating a Scenario is to define the Actions you would like to participate in the scenario and then save it.

Select the "Checkout Code" and "Deploy" actions in the left-hand box and click on the right arrow. This will move the actions to the right-hand box. Next, enter a name for the Scenario in the box above (we suggest: "Deploy") and click on Save.



This causes the Scenario to be created. However you still need to save the changes to the Component by clicking on the Save button at the bottom of the screen.

One thing you may have noticed is that there are Actions in the Available list that are not part of the Plugin. These actions include "Checkout Code", "Snapshot" and "Fetch". These are system actions and are provided by the Tableaux system, rather than by the Plugin.

Your HTML Files Component should now, by default, run the Deploy scenario.

Modify the Parameters

You will notice that the "Plugin Parameters" section now contains several fields; the same fields that we created for the Plugin.



These fields appeared because we included an action that uses them.

The parameters are used to tell the Plugin how to deploy this Component into an Environment. You'll notice across the top of the second half of the screen is a row of tabs.



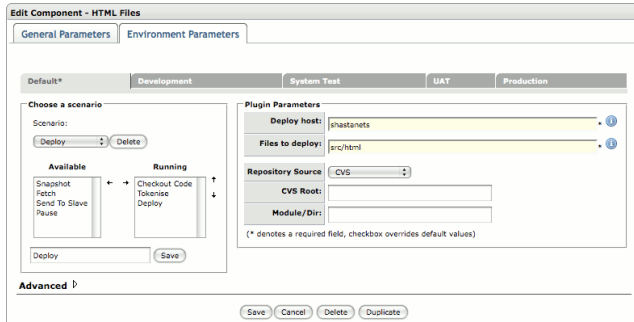
The environment tabs allow you to set parameter values for each environment. Click on each tab to bring it to the front.

The left-most tab is always called "Default". Values entered here will apply to all the environments, unless an environment overrides the default.

All other tabs have their fields greyed out and include an adjacent checkbox. Clicking on the checkbox activates the field and allows you to override the default value.



Go ahead and enter values into the fields in the default tab.



As you modify parameters the field will turn yellow to indicate which fields have changed. There is also an asterisk next to the tab name for any tabs that have un-saved changes.

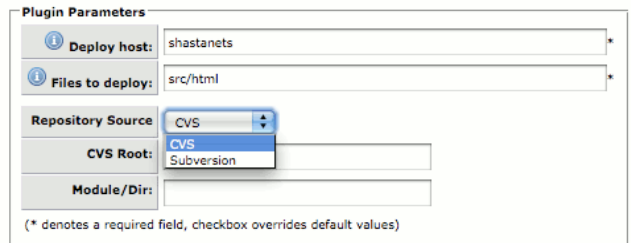
Navigating away from the screen with unsaved changes will cause the changes to be abandoned.

Choosing a Repository

Back in Chapter 3 - Before You Start, you installed a plugin that allowed Tableaux to talk to your particular Version Control System.

This made the *SCM* available to any Component that needs to extract files from it.

Since you have chosen a Scenario that includes the Checkout Code action, you'll notice that the Component Edit screen now provides a drop-down box in which you can select the repository type.



Depending on which *SCM* you choose, you will be presented with the requisite fields to identify where this Component will find its files.

For example, if you use *CVS* then you will be prompted for the *CVS Root* and *CVS Module* that contains the code you are extracting.

Enter the details and save the Component.

Deploying the Component

In the main Product work screen you'll notice that the HTML Files Component has a single icon in the cell for the Development environment. The icon is a picture of a cog with an arrow (⚙️).

This button allows you to deploy this component to this environment. Go ahead and click it.

The screenshot displays a deployment configuration interface with the following sections:

- Deployment Details:** Component: HTML Files, Environment: Development.
- Source artefacts:** Create new - Choose one of the following options:
 - Tag: Manually enter an existing Tag
 - Plugin does not support tag picking
 - Plugin does not support taggingDeploy existing:
 - Select an existing build
- Run Actions:** Scenario: Deploy, Actions: Checkout Code, Tokenise, Deploy. Includes a Modify button.
- Schedule:** When to deploy:
 - Now
 - 2010/12/12 17:10
 - Cron entry
- Options:** A dropdown menu.

Enter a tag into the Tag field (for example, "HEAD" for the head of the trunk), choose "Now" in the schedule box and click on the Deploy button at the bottom of the screen.

In the next chapter we will delve in depth into the deployment mechanism.

Part 3

Deployments

Deploying.....	40
Running a Deployment.....	41
Deployment Basics.....	43
Deployment Status.....	43
Deployment list.....	43
Deployment Comments.....	44
Mailing the Output.....	44
Halting a Deployment.....	44
Deployment History.....	44
Build Differences.....	45
Updating Last Status.....	45
Result Codes.....	47
Viewing Existing Result Codes.....	48
Creating New Result Code.....	48
Modifying/Deleting Result Codes.....	49
Component Plugins.....	50
Anatomy of a Plugin.....	51
Importing New Component Plugins.....	51
Writing New Component Plugins.....	51
Action Scripts.....	51
Scenarios.....	55
Plugin Life-cycle.....	56
Repository Plugins.....	57
Viewing Currently Installed Plugins.....	58
Installing/Upgrading Plugins.....	58

Deleting a Plugin.....	58
Configuring a Plugin.....	58
Tableaux's Internal Binary Repository.....	59
Using a Plugin.....	60
Plugin Aliases.....	60
Master/Slave Mode.....	61
Preparing Tableaux for Master Mode.....	62
Defining a Slave on the Master.....	62
Putting a Tableaux instance into Slave Mode....	62
Using Slave Mode.....	63
Release Kits.....	65
Overview.....	66
Viewing List of Release Kits.....	67
Release Work Screen.....	67
Creating a Release Kit.....	67
Deploying a Release Kit.....	69
Viewing Output.....	69
Release History.....	70
Implementation Plans.....	70
Single Release Mode.....	70
Master Plan Mode.....	71
Tokens.....	72
Format of a Token.....	73
Token Bundles.....	73
Creating a Bundle.....	73
Modifying/Deleting a Token Bundle.....	74
Creating a Token.....	74
Modifying Token Values.....	74
Assigning.....	75
Missing Tokens.....	75
Tokens Within Tokens.....	75
Standard System Tokens.....	76
Token History.....	76
Token Templates.....	77
Files Not to be Tokenised.....	77
Tokens in Component Parameters.....	77

Deploying

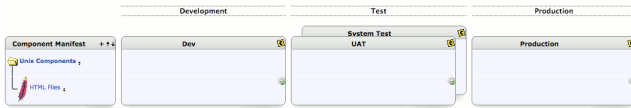
Deployments are the backbone of the Tableau system. "Deploying" is the process of creating a build and deploying it into a target environment.

Running a Deployment

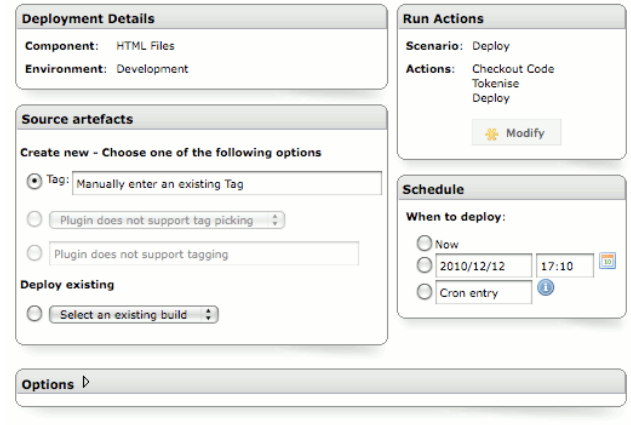
To execute a deployment you must first have the appropriate permission for the specific component and specific environment (see Chapter 22 - User & Group Management).

You can initiate a deployment from either the Product screen or from the main work screen.

The cell in the component/environment matrix will contain a deploy button (🚀) for each component you have permission to deploy.



Upon clicking the deploy button, you are taken to the deployment screen:

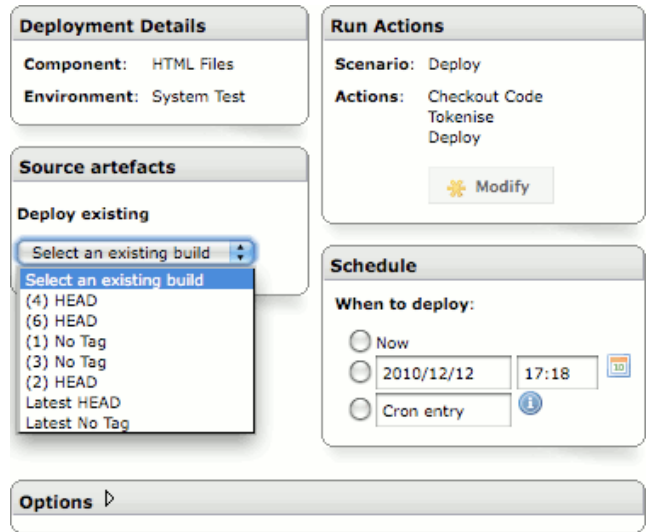


Your options on this screen depend entirely on whether the environment you are deploying to is a Base Environment or not.

- Base Environment - An environment into which you can introduce new builds (see Chapter 24 - Object Attributes & Permissions). Usually, most development environments will be base environments.
- Non-Base Environment - An environment in which you are *not* allowed to introduce new builds. Only existing builds can be deployed to non-base environments.

When deploying into base environments, you can choose any of the four options (as shown above), depending on the capability of the Repository Plugin.

For non-base environments, you can only use the Deploy Existing drop-down.



Creating a New Build

When creating a new build you are commonly extracting code from an SCM. However, version

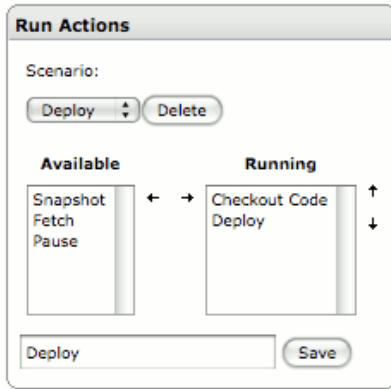
control systems have varying tagging systems. Tableaux will display all fields necessary to identify a particular tag or label, depending on the SCM being used.

Deploying Existing Build

You can choose from only existing builds that have successfully been deployed into other environments. Note: A build must have gained a green tick (✔) before it can be deployed into other environments.

Run Actions

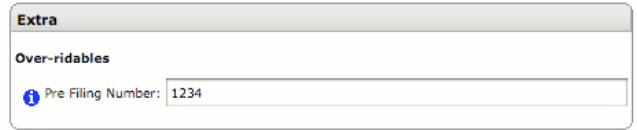
The run actions reflect the scenario that has been set up for this component. You can perform a one-time change of the run actions for this deployment only. Click on the Modify button and add or delete actions.



Over-ridable Fields

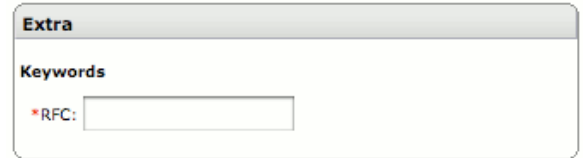
Some plugin parameters can be marked as over-ridable at runtime (see Chapter 12 - Component Plugins). If so, then you will be given an

opportunity to change the values here.



Keywords

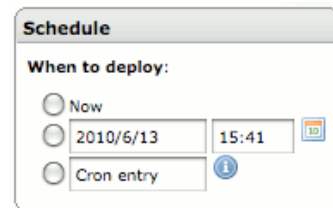
If keywords are required for this deployment then they will be displayed here. Enter a value for all mandatory keyword fields.



Schedule

You can schedule a deployment task to run at a later time. You may choose:

- **Now** - Kick the deployment off immediately.
- **Once off** - Kick the deployment off at a specified time in the future.
- **Recurring** - Make this a recurring job. Enter a cron-style entry to determine when to run.



See Chapter 32 - Scheduler for more information.

Extra Options

There are some extra options available to customise the environment. They are:

- Show extract files - Display a full listing of all files extracted from the repository. Otherwise, only the number of files extracted will be displayed.
- Enable deployment debugging - see later in this chapter.
- Use tokens - Use tokens from a past deployment. See Chapter 16 - Tokens.
- Promotion note - Allows you to enter a free-form text note that will forever be attached to this deployment.

Finally, click on the Deploy button to initiate the deployment.

Deployment Basics

When a Component is deployed, a temporary sandpit is created. This sandpit is created in a temporary directory that differs in location depending on your Operating System. For Unix OS's it is created in `/tmp`. For Windows, it's `c:\temp`. The location of this sandpit is customisable (see Chapter 26 - System Configuration).

All actions that TableauX takes are relative to the sandpit. For example, if code is exported from your SCM then it is exported into the sandpit.

After the sandpit has been created, the Actions are run in sequence until either an Action fails or TableauX runs out of Actions.

The sandpit is cleaned up after the deployment has finished.

Deployment Status


Each deployment has a status. This status is reported in the history screen. The status is determined by the aggregate of the Result Codes of all the actions that were run during the deployment.

The Result Codes are entirely customisable. You can create or delete Result Codes to support any logical outcome of a promotion.

For more information about Result Codes, refer to Chapter 11 - Result Codes.

Deployment list

When TableauX is instructed to deploy a component, it forks off and runs the deployment in the background. A screen is available to view the current list of deployments in progress. This screen is called "Deployment List" and is accessible from the main menu.

Component Deployments								
User	Component	Environment	Tag	Status	Started	Finished	Actions	
1 admin	HTML Files	Dev	HEAD	Finished 	Jan 3, 2010 1:37 PM	Jan 3, 2010 1:37 PM	Show Output	

The number of slots in the deployment list depends on the number of concurrent deployments defined in the *Global Configuration* (see Chapter 26 - *System Configuration*). New deployments appear at the bottom of the list, and old deployments drop off the top of the list.

You may attach to the output of any deployment, running or finished, by clicking on "Show Output" in the deployment list screen.

Deployment Comments

You are free to add comments to each deployment both before and after the deployment has occurred. Comments may be used to explain why a particular deployment failed, or what you did to fix it when it finally succeeded.

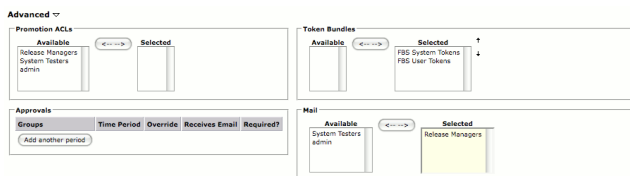
To add comments beforehand simply type them into the "Promotion Note" field in the Deployment screen.

To add comments to a deployment after it has occurred you must select (🗨️) from the history screen.

Mailing the Output

Occasionally you might need to send the output of a deployment to a person or group of people. This is especially useful for unattended deployments, such as those that have been scheduled to run in the future.

Each Component can be customised with a group or groups of users to send out an email, either by default or for particular Environments. You'll find the option to set the mail address under the Advanced drop-down on the Component Edit screen.



Here you can select the group(s) that will receive mail after the deployment has completed.

Halting a Deployment

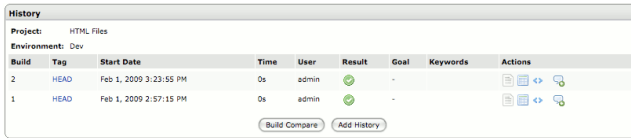
Occasionally you might need to stop a running deployment. For every current deployment there is an option in the Promotion List to stop the deployment.

From the menu bar, choose "View -> Promotion List". Click on the "Stop Promotion" link next to the appropriate entry. The deployment will be halted at the first available opportunity.

Deployment History

You can view the result of every deployment by clicking on the History icon (📖) in either the main work screen or the Product screen.

This brings up a screen such as the following:



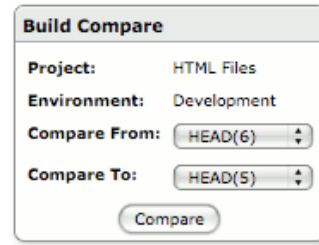
Here you can see information about each deployment and also perform the following:

- View the files from the SCM that were extracted by the deployment.
- View the output of the deployment.
- View the component parameters used during the deployment.
- View the tokens replaced during the deployment.
- View the deployment notes.
- Add a deployment note.

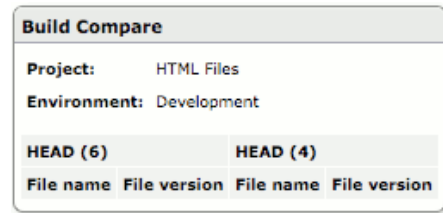
Build Differences

Tableaux can calculate differences at the file level between two builds of a component. This is useful to see which files (eg, in CVS) have changed from one tag to another, or even to see if developers have snuck in file changes and re-tagged them under the same tag.

There's a button available in the Component History screen called "Build Compare" used to initiate build differences. Clicking on this button results in a screen where you can choose which two builds to compare:



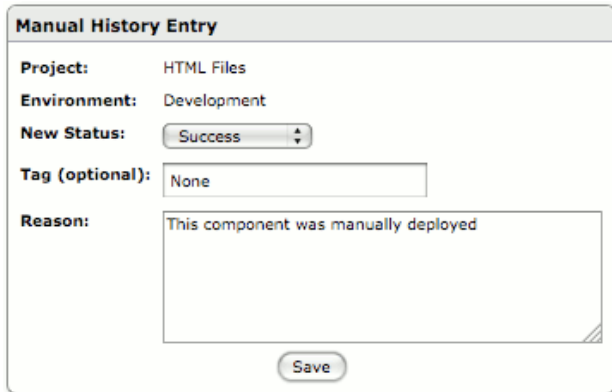
Choose two builds and click on the Compare button. A screen is displayed showing the differences between the two builds:



Updating Last Status

Tableaux also includes the ability to insert a dummy deployment result into the history. This is useful in many scenarios. For example, if a deployment is performed manually, then a dummy deployment result can be entered to reflect the actual environment. Likewise, if a component is made obsolete, it is possible to indicate this in Tableaux.

In the History screen (see above), click on the Add History button at the bottom of the screen.



The image shows a dialog box titled "Manual History Entry". It contains the following fields:

- Project:** HTML Files
- Environment:** Development
- New Status:** A dropdown menu with "Success" selected.
- Tag (optional):** A text input field containing "None".
- Reason:** A text area containing "This component was manually deployed".

A "Save" button is located at the bottom center of the dialog box.

Choose the status of this history entry (Passed, Failed, Not Sure, etc), and enter a note as to why the entry is being created. Optionally, a tag can be entered that will appear in the main work screen. This is used in circumstances where the history item relates directly to a tag, such as if a particular version of the component is obsoleted.

Result Codes

Every Action you create in the Plugins is linked to a script. As you execute an Action, the script is run, which eventually finishes with various return codes to indicate the final status to Tableaux. You have the opportunity to specify these return codes both within the scripts and within Tableaux.

By specifying a return code in Tableaux, you can determine how Tableaux reacts to the execution of the script.

When just a single Action is run during a deployment, the overall result of the deployment exactly matches the result of the single Action. When multiple Actions are run during a deployment, a system determines the final result code based on escalating levels of severity.

This chapter delves into how to create new result codes to match your Action scripts, and how to determine the final result of Tableaux deployments.

Viewing Existing Result Codes

Log into the system and select "Administration -> Result Codes" from the menu.

The existing result codes are displayed in a list.

Name	Description	Result Code	Return Code	Halts Prom	Superseded Level	Next Level	Colour	Graphic	Delete
Deprecated	Deprecated components	D	-2	<input type="checkbox"/>	10	<input type="checkbox"/>	CCCCCC	status_P.gif (-)	Edit Delete
Failed	Failed promotion	F	-1	<input checked="" type="checkbox"/>	10	<input type="checkbox"/>	FF0000	status_F.gif (-)	Edit Delete
NotSure	Success	N	10	<input type="checkbox"/>	5	<input checked="" type="checkbox"/>	FF00FF	status_N.gif (-)	Edit Delete
Success	Successful promotion	P	0	<input type="checkbox"/>	1	<input checked="" type="checkbox"/>	00FF00	status_P.gif (-)	Edit Delete
Warning	Success with warning	W	5	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>	FF00FF	status_W.gif (-)	Edit Delete
				<input type="checkbox"/>		<input type="checkbox"/>			Create

There are three default Result Codes. They are:

- **Passed** <✔> - A component that deployed without any errors.
- **Failed** <✘> - A component that incurs an error on any action. This prevents a build from being deployed to any other environment.
- **Unknown** <?> - Sometimes TableauX cannot automatically deduce the result of an action (which may require manual verification). This result can be changed later, if required.

There are many fields on this page that require explanation:

- **Name** - A single word description of the code.
- **Description** - Description of the code.
- **Result Code** - A single character representation of the code.
- **Return Code** - The code returned from the action script that invokes this result status.
- **Halts Prom** - If an action returns this result code which has this tickbox checked, then the promotion will be halted immediately.

- **Superseded Level** - The superseded level indicates the level at which another return code will supersede this result code. The total result code for a promotion will essentially be the result code with the highest superseded level.
- **Next Level** - Indicates whether a deployment with this status is a candidate to be promoted to the another environment.
- **Colour** - Indicates the 3 byte colour code for this result code.
- **Graphic** - Indicates the graphic to use in the work screen.

Creating New Result Code

From the menu bar, choose "Administration -> Result Codes". The bottom row of the table contains empty fields in which you can enter details for a new result code.

Enter the name and description. Please note that the "Result Code" field must be a unique single character.

In the Return Code field, enter a number that matches the return code that an action script will throw. An action script may throw many different return codes depending on its degree of success or failure.

- All Unix and Windows operating systems allow return codes in the range 0 - 255.
- The "-1" return code is used by TableauX as a catch-all and is by default assigned to the Failed result code.

- If you set any other negative number, then the Result Code can never be reached as a result of an action, and is usually reserved for manual purposes.

The Superseded Level indicates another return code that will supersede this one. This effectively defines a severity level ranking.

There is a colour palette icon in the Colour field to help you choose a colour.

New graphic icons can be added to Tableaux by copying a .gif or .png file to `<install dir>/tomcat/webapps/tableaux/images`.

Modifying/Deleting Result Codes

To modify or delete an existing result code, you must first bring up the appropriate screen (from the menu bar, choose "**Administration -> Result Codes**").

Click on either the Edit or Delete button next to the code you wish to modify or delete.

Any changes take effect immediately.

Component Plugins

Earlier on we briefly addressed the topic of creating a plugin. This chapter will expand on this topic, as it is the most critical to the Tableaux system.

Tableaux is a build and release management system. How it performs these tasks is defined by the component plugins. Each component defined in Tableaux is associated with a plugin. This plugin knows the processes required to compile and deploy the contents of the component, or indeed, run any type of action for the digital artefact.

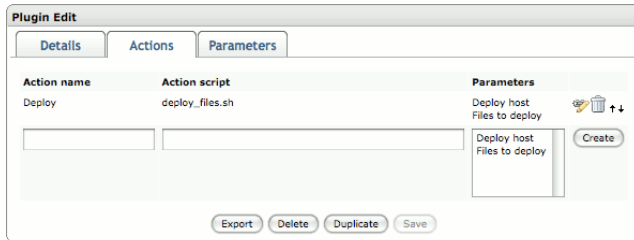
The screenshot shows a 'Plugin Edit' window with the following fields and controls:

- Name:** Deploy HTML
- Description:** Deploys static content to the web server
- State:** Working
- Image:** pp_apache.gif (with a pencil icon and '- (upload new)')
- Buttons:** Export, Delete, Duplicate, Save

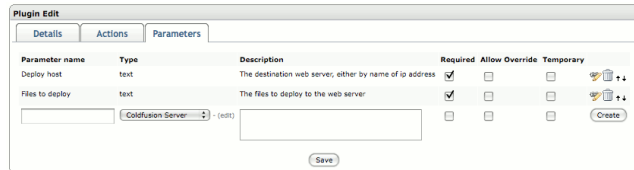
Anatomy of a Plugin

As shown in Chapter 8 - Creating Your First Project-Plugin a component plugin consists of a set of actions and a set of parameters.

The actions encapsulate all the operations that a plugin can perform. Each action is associated with a script to run.



Each Action also has associated parameters. The values of these parameters are passed into the actions for them to perform their tasks. For example, a "file-deploy" action may require a "Server" parameter to send the files to.



Importing New Component Plugins

Log into the system and select "Administration -> Project-Plugins" from the menu.

Select the import button and enter the file name of the downloaded file (Tableaux plugins have the extension .txp). The plugin will be imported into

the system and is available for use immediately. However it is a good idea to check each of the scripts and make sure you understand what they are doing. You may need to customise the scripts to suit your environment. Refer to the following sections of this chapter for information on how to do this.

Writing New Component Plugins

One of the flexible features of Tableaux is the ability to create your own plugins and to customise the existing ones.

Creating new plugins was briefly covered in Chapter 8 - Creating Your First Project-Plugin.

The remaining sections of this chapter will delve into more detail about each aspect of the plugin.

Action Scripts

The location of the action scripts is defined in the Global Configuration (see Chapter 26 - System Configuration).

If you are creating a new component plugin then you can place your scripts beneath this directory, since all scripts are referenced relative to it. It is a good idea to create a sub-directory to contain the scripts for each plugin. For example:

```
<tableaux dir>/scripts/plugin1
<tableaux dir>/scripts/plugin2
<tableaux dir>/scripts/plugin3
...
```

The action scripts were briefly covered in Chapter 8 - Creating Your First Project-Plugin. This chapter will provide advanced coverage of the topic.

Action Script Rules

In general, action scripts are left up to you to determine what they do, however there are just a couple of rules that they must follow. They are:

- You may write your scripts in any language. They may even be compiled programs, such as Java or C. Using interpreted scripts (such as .bat files on Windows or Shell scripts on Unix (or Groovy scripts on either platform - see below) allows for much easier debugging.
- The scripts will be run as the owner of the servlet engine running Tableaux. By default this is the "tableaux" user. Thus you may find "sudo" handy where you need to run things as a different user.
- The scripts are run without a controlling terminal and therefore will not block to wait on user input. Please ensure that any program expecting user input does not block. Many programs have command line parameters to get around this, or else you may find the *Expect* program useful (on Unix) where there aren't any, especially for programs like *password* that go to the terminal for input.

- The scripts are provided a temporary directory, or sandpit, in which they can do their work. You should not unintentionally allow scripts to write any temporary files outside of this directory. You may leave the files in this directory as the directory is cleaned up automatically by the Tableaux framework after the script finishes.

Action Script Result Codes

Each action script returns with an exit code, according to the Operating System it is running on. Tableaux provides a look-up table for exit codes to determine how to proceed after receiving particular codes from the scripts. This look-up table is described in Chapter 11 - Result Codes.

Action Script Skeleton

A template action script is provided for you in the distribution called `template.sh` for Unix and `template.bat` for Windows.

Have a glance at these templates right now, as the following sections will be much clearer.

Action Script Life Cycle

Each script is passed four (4) parameters, regardless of the Operating System you are running on. These parameters are:

- The location of the sandpit containing any code exported from the SCM.
- The path to a file containing the component parameter values for this environment.
- The name of the action being run.

- The sub-path to any checked out code.

The second parameter passed into an action script is the location of a "parameter file" containing the parameter values defined in the component. There is an application provided to allow you to extract these parameters from the file. This application is called 'get_param' and takes two parameters. The first is the parameter file passed into your main script as its second parameter. The second is the name of the parameter defined in Tableaux:

```
java get_param <param file> <param name>
```

For example:

```
java get_param $2 "Deploy host"
```

There is also a shell script version of get_param that is considerably faster, but does not handle textarea parameters. You can use it like this:

```
get_param.sh $2 "Deploy host"
```

Finally, there is also a script available to make the token values available to the action scripts. This script is called "get_token.sh" and is used in exactly the same manner as get_param. The only difference being the second parameter is the token name to extract instead of the parameter name:

```
get_token.sh $2 <token name>
```

The general layout of your action script will be to:

- Fetch any parameter values you need
- Perform the work (compile, deploy, etc)

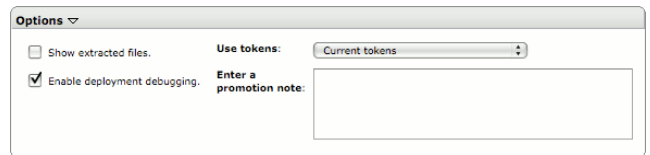
- Return with an appropriate exit code.

Debugging Action Scripts

When writing your action scripts, they will rarely perform exactly as you require out of the box. Some debugging is usually required to get them going.

Tableaux includes a mechanism to help you debug your scripts.

In the deployment screen (see Chapter 10 - Deploying) there is an option under Options to enable debugging.



When ticked, this sends an extra parameter to your action script that you can use to determine when to turn debugging on.

On Unix, you would include the following snippet in your deployment scripts:

```
# Enable debugging
DEBUG=`java get_param "$
{CONFIG_FILE}" \
"Enable Debugging"`
[ "${DEBUG}" = "on" ] && set -x
```

On Windows you would include the following:

```
rem Enable debugging
for /F "usebackq delims==" %i in (`
%JAVA_HOME%\bin\java.exe -classpath
```

```
%PARAM_DIR% get_param %CONFIG_FILE%
"Enable Debugging"`) do set DEBUG=%i

if "%DEBUG%" == "on" echo on
```

This causes deployments to output extra debugging which allows you to figure out most problems.

Note: Enabling debugging for a promotion may expose passwords into the output stream. For this reason, only users with access to run a debugging deployment are allowed to view the output from one. This means that debugging access should be a very restricted right.

Groovy Action Scripts

Tableaux includes an embedded Groovy script interpreter. Groovy is an interpreted language that is very close to Java. It allows you to use Java libraries in interpreted scripts.

For information about Groovy and the language syntax, refer to the Groovy home page: <http://groovy.codehaus.org/>

The embedded interpreter means that you can write your action scripts in pure Groovy. Groovy scripts differ slightly from shell scripts, but the principle is reasonable similar. Any action script ending in `.groovy` is deemed to be a groovy script and is sent to the Groovy interpreter rather than the Operating System for shell execution.

A skeleton groovy script is already included in the Tableau distribution called `template.groovy`. It is reproduced here:

```

/*****
// The parameters passed in are:
//   workingDir - The temporary
Tableaux working directory
//   params      - The properties
object containing all the project
parameters
//   action      - The name of the
action currently being run
//   relativeDir - The relative
directory that gets checked out. Eg,
for CVS this is the equiv. to
CVS_MODULE
/*****
/*****
// In return, you must define the
variable "output"
output = ""
/*****

/*****
// Now fetch the project parameter
values for this plugin
def dest_dir =
params.getProperty("Destination dir")
def zip_file = params.getProperty("Zip
file");
/*****

/*****
// Now actually run the plugin. Eg:
output += "Parameters passed:\n"
output += "  Working Directory: $
{workingDir}\n"
output += "  Relative Directory: $
{relativeDir}\n"
output += "  Action: ${action}\n"
output += "\nProject parameters:\n"
output += "  Destination dir: $
{dest_dir}\n"
output += "  Zip file: ${zip_file}\n"

return 0;

```

System Actions

There are several actions provided by the system, rather than by the plugins. They are:

- **Checkout Code** - Forces Tableaux to export code from the specified repository. Extra fields are displayed allowing you to enter the details of the repository.
- **Tokenise** - Scans the working directory for files that contains "tokens". Replaces those tokens with values from the database.
- **Snapshot** - Forces Tableaux to snapshot part or all of the sandpit working directory. This is useful to capture binaries after they have been compiled.
- **Fetch** - Restores the working directory as it was captured by the Snapshot action. This is useful in later environments where you want to promote the binaries that were compiled in earlier environments.
- **Mail** - Causes Tableaux to send out an email after the deployment has completed providing the details and status of the deployment.
- **Pause** - Will pause the deployment for the specified amount of time.

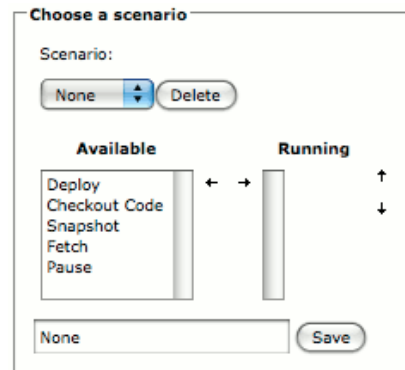
Scenarios

In Chapter 9 - Bringing it all Together we introduced Scenarios. Scenarios exist to capture pre-defined execution plans for your plugins. You can create any number of scenarios for a Plugin. Any scenarios you create become available to all other components sharing that Plugin.

Creating a Scenario

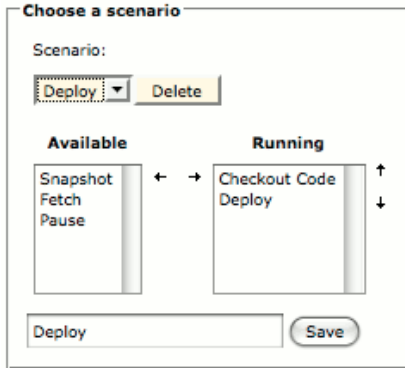
To create a Scenario, you'll want to edit a Component that uses the plugin.

The left-hand side of the Component edit screen is dedicated to Scenario creation. Initially there will be no Scenarios available (as indicated by the None in the drop-down box).



The premise of creating Scenario is to define the Actions you would like to participate in the scenario and then save it.

Select the Actions you need in the Scenario and click on the right arrow icon to move them from the Available box to the Running box.



Use the up and down arrow buttons to organise the running order of the actions. Enter a name for the Scenario in the box above and click on Save.

Deleting a Scenario

To delete a scenario, you'll want to edit a Component that uses the plugin.

Select the scenario in the "Scenario:" drop-down box and click on the Delete button.

Plugin Life-cycle

Each plugin has a "State". You can change this state at any time to any of the following:

- **In Development** - This is the initial state of a new plugin. Only users in the selected groups can deploy components that use this action.
- **Working** - This should be the default state for a working plugin. All users can deploy components based on this action.
- **Restricted** - This plugin is working, but its use is restricted to users in the selected groups.

- **Locked** - No components can be deployed that use this plugin. This is usually a temporary state.
- **Deprecated** - The plugin is no longer in use. No components can use the plugin.

The plugin should be in development state whenever work is being done in its deployment scripts. This ensures no components are accidentally deployed using the plugin.

The plugins should be in the "Working" state by default.

Repository Plugins

Most Components will use Component-Plugins that call an Action "Checkout Code" to extract files from a version control system; whether it be source code, board diagrams, documentation, etc. This Action can interface with numerous version control systems thanks to Tableaux's repository plugin architecture. New and upgraded repository plugins may be installed as and when they become available.

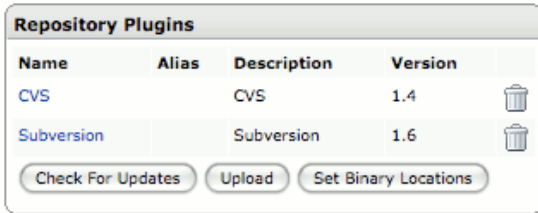
This chapter describes only how to un/install plugins to repositories. For repository specific notes, and specifically on how to configure them, please refer to the online documentation available at:

http://www.incanica.com/doc/tableaux/repository_notes.html

For this chapter only, the bare term "plugin" refers to a Repository Plugin rather than a Component Plugin.

Viewing Currently Installed Plugins

To see which plugins are currently installed, select “Administration -> Repositories” from the menu.



Plugins are identified by their name and a version.

If your repository is not listed and it is an officially supported SCM then you will need to download and install the plugin.

Installing/Upgrading Plugins

There are two methods to install plugins into TableauX. Incanica maintains a repository of plugins available over the Internet. From the repositories screen, click on the Check For Updates button. This button searches the plugin archive for plugins that are compatible with your version of TableauX.

Plugin Name	Installed Version	Available Version
StarTeam9	1.4	Install
StarTeam	1.4	Install
Hudson 1.X	1.1	Install
HTTP File	1.1	Install

You can click on Install or Update accordingly to download and install the specified plugin.

Alternatively, if you have the plugin as a file on a disk, then you can upload it directly into TableauX. From the Repositories screen, click on the Upload button.

Type the location of the file, or click on the Browse button to locate the file. Then click on Upload.

Deleting a Plugin

To delete a plugin, simply click on the trashcan icon (🗑️) next to the plugin name on the Repositories screen.

Configuring a Plugin

Most plugins require some degree of configuration before they will work.

To configure a plugin, click on its name in the Repositories screen.

Most plugins, at a minimum, require you to enter credentials for a user for TableauX to connect to the repository.

Tableaux's Internal Binary Repository

The system-defined Snapshot action is used by Tableaux to store off any digital artefacts that you want to keep before Tableaux cleans up the temporary deployment sandpit.

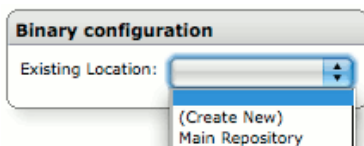
The most common use-case is to store off any compiled binaries for deployment into later environments.

When the Snapshot action is called during a deployment, Tableaux stores the entire defined directory into a repository. Any repository that Tableaux has a Plugin for can be used to store the artefacts.

You must first create a repository in the SCM according to that software vendor's instructions and provide a user with full read/write permissions to that repository.

To avoid any potential security hazards, we strongly encourage you to provide Tableaux with its own, dedicated, repository for Tableaux's digital artefacts rather than re-use an existing one.

Select "**Administration -> Repositories**" from the menu. Click on the Set Binary Locations button.



You can either create a new repository definition or edit an existing one.

Enter in the details for a repository. The details typically include the location of the repository and the credentials used to connect to it (note: remember that Tableaux will not automatically create the repository in the SCM - you will have to do that yourself according to your SCM vendor's instructions).

Once you have entered the values click on the Save button.

Now that a binary location has been defined, you can set it as the default. See Chapter 26 - System Configuration under the Miscellaneous section.

Tableaux can be configured with multiple such binary locations. If required, each Product can override the default and store its artefacts in a different location.

To have a Product store its artefacts in a different location, first define the location (as described above). Next, edit the Product (see Chapter 7 - Creating Your First Product). At the bottom of the Product Edit screen, you can change the location its binaries are sent to.

Using a Plugin

In Chapter 12 - Component Plugins we introduced the *Checkout Code* action during deployment.

This action calls upon a repository plugin to interact with the *SCM* and extract the files for the deployment.

When a component uses the *Checkout Code* action, you are presented with a drop-down box with the available repository plugins.

Once you select a repository source, there are several component parameters presented with *SCM*-specific fields for you to fill in. Typically these allow you to define the location of the repository, etc.

These fields change depending on the repository plugin you have chosen.

Plugin Aliases

Often the name of a plugin will include its version. For example, "StarTeam 10".

In the previous section "Using a Plugin" each component is set up with the plugin name.

If you were to upgrade the StarTeam plugin to "StarTeam 11" in the future then all existing Components will be configured incorrectly.

To avoid this problem you can "alias" the repository name. If you upgrade the plugin in the future you can move the alias to the new Plugin without having to change each and every Component that uses that Plugin.

Master/Slave Mode

The Action scripts (detailed in the previous chapter) are normally written to use best-practice connection methodologies, depending on your environment and requirements. For example, using SSH is common to connect to other servers.

Sometimes a connection is not possible to make from the Tableau server using existing connection methods. For example:

- Connecting from Unix to Windows is difficult where no SSH server is available on your standard Windows platform
- Connecting into segregated LANs such as Dev/Test or DMZ LANs restricts where you can connect to.
- There is no remote client available for a 3rd party tool

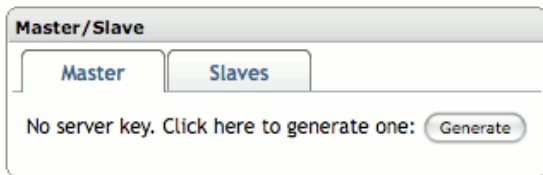
To cater for these scenarios, a Tableau instance can be forced to operate in a Slave mode, acting as a remote agent that can receive deployment requests from an authorised Master Tableau instance.

Your security is assured because the Slave Tableau instance only runs with enough privileges to service the deployment requests, and the connection between the Master and Slave Tableau is made using best-of-breed encryption technologies to guarantee that no eavesdropping can occur nor unauthorised connections be made.

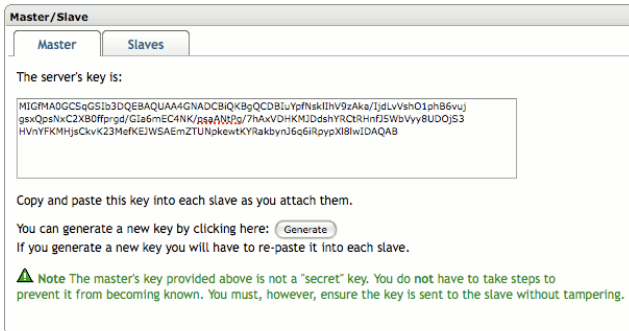
Preparing Tableaux for Master Mode

Your Master Tableaux instance can be any existing Tableaux server.

Log onto your production Tableaux server as a privileged user and from the menu select **"Administration -> Master/Slave"**.



Click on the **"Generate"** button. This will generate a new Master's key.

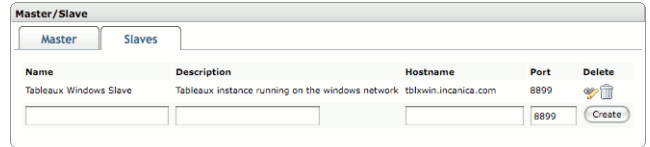


This key is a **"public"** key. It is not important to keep it secure, however it is important that it is sent to the Slaves without modification.

You will need to copy and paste this key into each Slave that you create on your network.

Defining a Slave on the Master

The Master Tableaux instance must be told about all of the Slave instances on your network. On the Master/Slave screen, click on the **"Slaves"** tab.



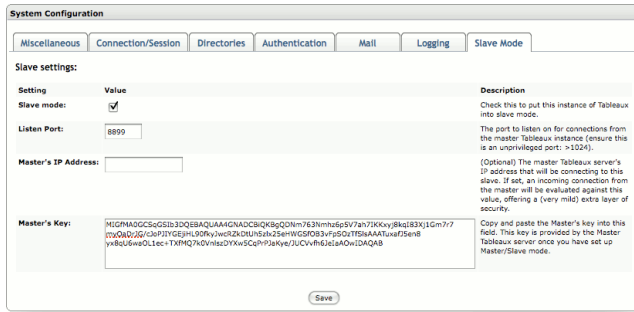
You can create or edit any number of Slave instances on this screen. Ensure each Slave has a unique **"name"**.

The default port for each Slave is 8899, however you can change this to any unprivileged port (usually above 1024).

Putting a Tableaux instance into Slave Mode

The Slave Tableaux instances should be virgin Tableaux installations. Install Tableaux according to the instructions in Chapter 1 - Installing Tableaux.

Log into this instance as a privileged user (usually **"admin"**). From the menu select **"Administration -> System Configuration"**, and choose the **"Slave"** tab.



To put the Tableau instance into Slave mode, you will need to tick the "Slave mode:" checkbox.

If required, you can change the port the Slave will listen on. The default is 8899.

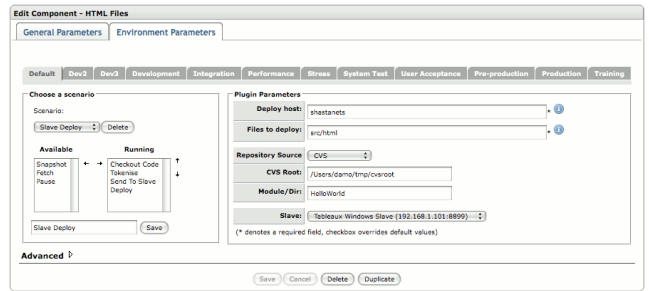
You can optionally enter the Master's IP address which will be compared against any incoming connections. This is not full-proof protection, as it is possible to spoof IP addresses.

Copy the Master's key and paste it into the last field.

Click on the Save button. You will notice that the main menu will change. The Tableau instance is now in Slave mode and does not allow general usage. It can only be driven by the authorised Master.

Using Slave Mode

When deploying a component, you can now use the "Send to Slave" system action.



There's two items of note in the screen shot above:

- The deployment scenario includes the "Send to Slave" action.
- The action provides a new drop-down to choose which Slave to send the deployment job to.

The deployment job starts off on the Master server as a normal job. When it hits the Send to Slave action, the following occurs:

- All the deployment scripts are synchronised over to the Slave
- The Master server packages up the deployment job and sends it to the slave. The deployment job includes:
 - The temporary deployment sandpit (including any potentially checked-out files)
 - All the meta-data surrounding the deployment job include: deployment parameters, tokens, etc.
- The Master instructs the Slave to continue the deployment job. The Slave will perform the actions *after* the "Send to Slave" action.

- Once the Slave has finished the deployment, it transfers control back to the master and sends back all the information about the deployment.

Validating component parameters...

Status: 

Action [1]: Sending To Slave: 192.168.1.101

Negotiating a secure connection...

Syncing the deployment scripts...
Syncing the configuration file...
Syncing the job details...
Syncing the sandpit...
Syncing the metadata...
Flushing the slave's cache...

Sync took 20s

Handing control over to the slave...

Slave server: 192.168.1.101

Action [1]: Export code...

Checking out files...
cvs_root: /Users/damo/tmp/cvsroot
cvs_module: HelloWorld
Checked out 6 files

Status: 

Action time: 1 second

Some things to note are:

- A deployment job can only be sent from the Master to the Slave. It is not possible to run more actions on the Master once this has occurred.
- If the Slave performs the Checkout Code action, then it is also the Tableaux instance that re-tags the repository files.
- If the Slave performs the Checkout Code action, then it must be configured with all required Repository Plugins (see Chapter 13 - Repository Plugins). The Repositories Plugin menu option is still available (even in slave mode) on the main menu for this reason.

Release Kits

So far we've discussed the deployment of individual components. As projects scale up to scores or hundreds of components, it becomes a herculean effort to individually track and deploy these components across multiple environments.

To make it easier to track these components, we'll introduce you to some new functionality called Release Kits. Release Kits provide a method to chain together the deployment of multiple components. Essentially, a Release Kit reflects the steps in an Implementation Plan, and can actually be used to produce the Implementation Plan automatically.

Release Kits provide many features:

- Automatically generates Implementation Plans
- Centralised run-sheet during deployment. Everybody on the same page.
- Reduces over-all release time by reducing the dead time between component deployment
- Invest time once, reap benefits many times
- Reduces chance of mistakes

Overview

A release kit looks like the following:

The screenshot shows a web-based configuration form for a release kit. The title is 'Release Kit: Flight Booking System - Full'. There are four tabs: 'Details', 'Dependencies', 'Tokens', and 'Steps'. The 'Details' tab is active. The form contains the following fields:

- Product:** Flight Booking System (dropdown menu)
- Name:** Flight Booking System - Full
- Version:** 2.5rc1
- Components:** (empty text area)
- Platform:** Linux, Apache
- Creator:** admin
- Business Link:** http://cmdb/object?id=1234
- Summary:** Deploys the entire FBS application, including database
- Promotion Notes:** (empty text area)

At the bottom of the form are three buttons: 'Save', 'Delete', and 'Generate Implementation Plan'.

Entities contained in a release kit:

Details

- **Product** - Optionally assigns this release kit to the named Product.
- **Release Name (mandatory)** - The name of the release kit.
- **Version (mandatory)** - The version of this release kit.
- **Components** - A summary of the list of components this release kit encompasses.
- **Platform** - The technical platform (operating systems, languages, etc) used by the release kit.
- **Creator**
- **Business Link** - A link (URL, or otherwise) to this application in any business systems (eg, a CMDB system).
- **Summary** - A free-text summary of what this release kit does.
- **Promotion Notes** - Some brief notes about the deployment process.

Only the name and version are mandatory fields. The others are optional.

Dependencies

Dependencies record the pre-requisites for the release (both internal and external to Tableaux). Dependencies can be one of:

- **Internal Component:** This is a component in Tableaux that this release depends on.
- **Internal Release:** This is a Release Kit that this release depends on.
- **External:** This is an external dependency. Tableaux cannot enforce this, it is merely for the record.

Tokens

This section holds any changes that must be applied to tokens during the release process. Sometimes it may not be wise to update token values until the time of the release.

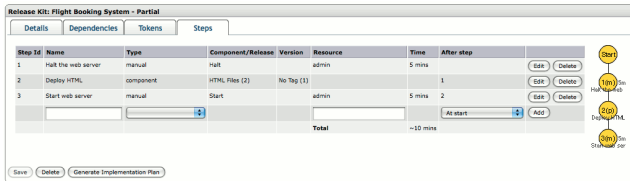
Execution Steps

This section contains the steps that are executed when the Release Kit is deployed. Steps can be one of:

- **Manual:** Defines a manual task that must be performed at this stage of the release. The release will pause until the task is complete. Only those people in the nominated group are able to continue the release.
- **Component Promote:** Defines a component to deploy within Tableaux.

- **Release Kit:** Defines another Release Kit to deploy. Similar to the component step, you must define which Release Kit and Version to release.

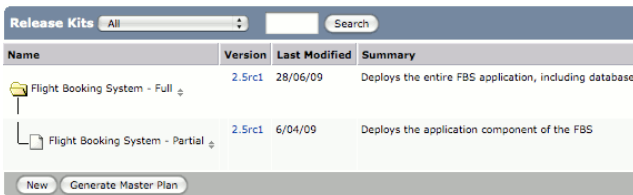
On the right hand side is a picture graphically showing the execution path of the release. This picture updates as you add or remove steps from the Release Kit.



Below the Steps is a total time to run. This is an estimate, based on past history, of how long the Release Kit will take to deploy.

Viewing List of Release Kits

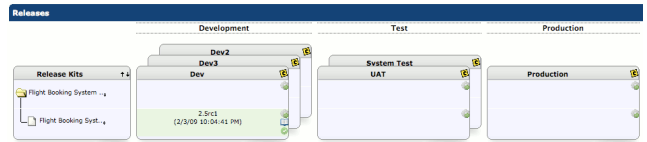
Select **"Release -> Release Kits"** from the menu to bring up the list of release kits.



The release kits are organised in a hierarchical manner, similar to components. Only the latest version of each Release Kit is visible by default. You can see all the versions of a release kit by clicking on the triangle next to the version.

Release Work Screen

The release work screen is the equivalent of the project work screen. You can access the Release Work Screen by clicking on **"View -> Release Work Screen"** from the main menu.



This screen displays the latest versions of each Release Kit in each environment. As with the Component Work screen you can access the history for each environment, as well as deploy a release kit into a specific environment.

Creating a Release Kit

There are 5 ways to create a Release Kit.

From the Release Kit screen

Open the List of Release Kits by selecting **"Release -> Release Kits"** from the menu. At the bottom of this screen, click on the New button.

From this screen you can build up a release kit from scratch.

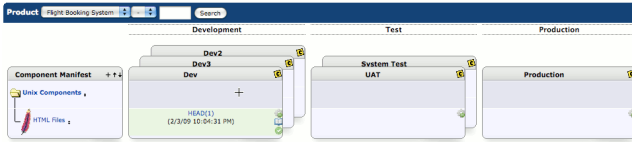
From an Existing Release Kit

Navigate to an existing Release Kit. After making changes to it, you would normally just click on the Save button. However, if you modify the name and/or version of the Release Kit it will be saved as a brand new Release Kit.

From a Group of Components

Enter the project work screen by selecting **"View -> Project Work Screen"** from the menu. Navigate to a group of components that you wish to create a release kit from.

From the main menu, select **"Actions -> Create a Release"**. The cells for all groups in each environment will change colour, and your mouse cursor will change into a cross when you pass it over the cell.

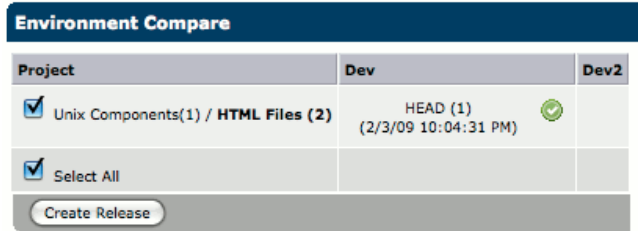


You can choose a group/environment combination and click on the appropriate cell. Tableau will auto-generate a Release Kit based on all the components below the group in that environment.

Enter the new release kit name and version and click on the Save button.

From Environment Differences

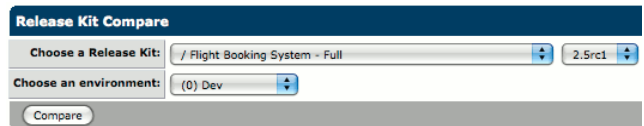
Follow the steps above for creating a Release Kit based on a group of components. However, from the menu choose the **"Actions -> Compare Two Environments"** option. This option will display the differences between two environments on a component by component basis.



Click on the Create Release button to generate a Release Kit out of the selected components. Once again, enter the name of the Release Kit and the version and click on the Save button.

From Release Kit Comparison

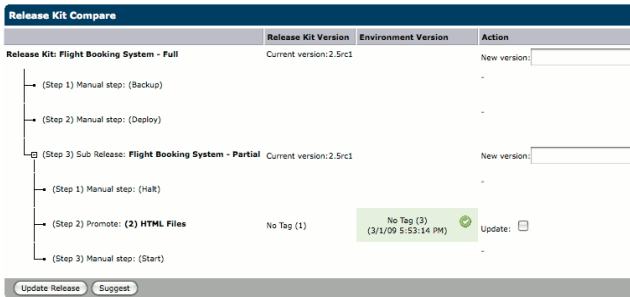
From the main menu, select **"Release -> Release Kit Compare"**. This screen lets you compare a Release Kit (and the versions of all the steps) to an environment.



This function is useful both to see how an environment has deviated from a baseline established by a Release Kit, and also to update a Release Kit based on changes made to an environment. This scenario is especially useful in a development environment which has had components promoted manually into it. By comparing a Release Kit, you can easily tell which components have been updated and easily bring the Release Kit up to date with any new changes.

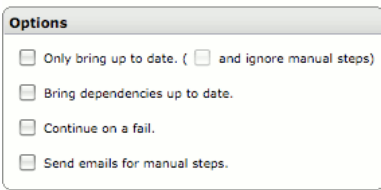
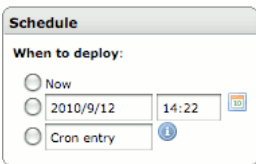
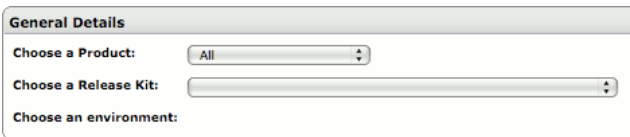
Choose a Release Kit and an Environment and click on Compare.

From the resulting screen you can enter new versions of the Release Kits to create. Check any checkboxes and enter new version names and click on Update Release. New Release Kits will be created and you will be given a summary of all actions taken.



Deploying a Release Kit

Deploying a release kit is a little different to deploying a component. Select **"Release -> Release Kit Promote"** from the main menu.



From this screen you must select:

- A product
- A release kit
- A release kit version (displayed after selecting a release kit)
- An environment

Some non-mandatory options are:

- There may be some keywords you will have to fill in.
- Schedule a release to run now, at a later date or at a recurring time.
- You can choose to only bring an environment up to date. This forces any components that already exist in the environment to be skipped.
- Continue on fail. This will force a release to continue, even if one of the steps fails.
- Send emails. Checking this checkbox will cause Tableau to send emails to the group of people responsible for each Manual Step as it arrives in a deploying release kit.

Viewing Output

The Deployment List screen (**"View -> Deployment List"**) details all currently running Release Kits. You can select a particular release and view its output. If a release kit is paused on a manual step, waiting for confirmation that it has been completed, then a box will appear on this screen (as shown below).

The screenshot displays the Release Kit interface with four main panels:

- Summary:** Shows release details for 'Flight Booking System - Full / 2.5rc1' in the 'System Test' environment. It lists the approver as 'admin', the start date as '6/07/11 3:16 PM', and the total running time as '2 minutes'. The state is 'Running step: 1 (manual)'.
- Running Steps:** A table with columns 'Step', 'Type', 'Name', 'Time', 'State', and 'Actions'. It shows one step: '1 - manual: Backup Database 2m, 13s' in a 'Waiting' state with a 'Finished' button.
- Node Graph:** A circular dependency graph with nodes for 'Start', 'Backup Database', 'Deploy Database', and 'Halt the web server'. Arrows indicate dependencies between these steps.
- Release Kit Progress:** A table with columns 'Name', 'Version', 'State', and 'Output'. It lists 'Flight Booking System - Full' with version '2.5rc1' and state 'Running step: 1 (manual)'. Below this is a 'Pre-check' section with a list of status messages, all of which are 'Passed'.

The screenshot shows the 'Release Kit History' table with the following data:

Release Name	Version	Environment	Date Finished	Length	Who	Keywords
Flight Booking System - Partial	2.5rc1	System Test	1/03/09 6:16:28 PM	3 mins, 12 secs	Super User (admin)	Output
Flight Booking System - Partial	2.5rc1	Development	3/02/09 10:04:41 PM	1 mins, 45 secs	Super User (admin)	Output

Below the table are search filters: 'Date from:' and 'Date to:' (both dd/mm/yyyy), 'Name:', 'Env:', 'Keywords:', and 'User:'. A 'Num rows: 100' selector and a 'Search' button are also present.

You can refine the displayed history items using the fields below the history listing. You can also refine how many history items are displayed.

The history is also available from the Release Work Screen.

In the release kit output, you'll see 5 panels. They are:

- A quick summary of the deployment.
- The steps that are currently running in *this* Release Kit. Also, any manual steps in the RK will be shown, along with an estimate of when they should be actioned.
- The overall progress of the release. This includes a list of all the sub-release kits that are deploying in the overall release.
- A node graph that reflects the steps in the release kit. The graph will turn colours based on various events:
 - Orange - This step is currently running.
 - Green - The step succeeded.
 - Red - The step failed.
- The textual output of the release.

Release History

Tableaux keeps an audit of all Release Kits that have ever been released. Select **"Release -> Release Kit History"** from the main menu.

Implementation Plans

Once you have a release kit defined, you can get Tableaux to generate your Implementation Plan for you.

The plan can be generated in two different modes:

- Single release mode.
- Master plan mode.

Single Release Mode

The first mode generates a straight forward plan for a given release. Select **"Release -> Release Kits"** from the menu and choose a Release Kit to edit. Whilst in the Release Edit screen select the Generate Implementation Plan on the bottom bar.

Hitting this button generates an Excel spreadsheet that can be opened in OpenOffice or Microsoft Excel. The implementation plan is fully editable.

Step	Task Name	Duration	Start	End	Who	Tag
1	Backup Database - Manual task: Backup	5 mins	010309 6:28 PM	6:50 PM	admin	
2	Deploy database - Manual task: Deploy	5 mins	6:50 PM	6:55 PM	admin	
3	Deploy HTML	5 mins	6:50 PM	7:00 PM	admin	
3.1	Start the web server - Manual task: Host	5 mins	6:50 PM	6:55 PM	admin	
3.2	Deploy HTML	5 mins	6:55 PM	6:55 PM	admin	No Tag(1)
3.3	Start web server - Manual task: Start	5 mins	6:55 PM	7:00 PM	admin	
	Finish:		010309 7:00 PM			

Similar to the single implementation plan, once you click on the "Generate" button you'll be given an Excel spreadsheet.

Note that if you change the "Start Time" field then that change cascades through all the timings on the spreadsheet.

If there is a job waiting for approval or currently scheduled, then Tableau will plant that scheduled time into the Implementation Plan as the starting time for the release.

Master Plan Mode

The second mode is a "master plan" mode, whereby several releases can be strung together on a single plan, each release scheduled to start at a particular time. Select **"Release -> Release Kits"** from the menu and click on the **Generate Master Plan** button at the bottom of the screen.

You will be taken to a page where you may add releases to be run at particular times.

Master Plan Itinerary

Release	Version	Time
/ Flight Booking System - Full	2.Src1	2009/02/28 12:00

You may add and delete as many releases as you like. The ordering that you place the Release Kits on this screen is the ordering in which they'll appear in the master plan. I.e, the master plan does no sorting (chronological, or otherwise) on this list of Release Kits.

Tokens

Tokens are an extremely easy concept to understand, yet provide Tableau with much of its flexibility

Of times there are differences between environments that cannot be addressed by promotion-time scripts. An example is a properties file inside a .jar file containing the hostname for a client to connect to. This hostname requires a different value depending on the environment of the build.

Tokens exist to cater for these differences in the environments.

In essence a token is simply a marker in the source code that is parsed by Tableau when the source code is extracted from the source code control system. It is replaced with a user-defined value particular to the environment the project is being deployed into.

Format of a Token

A token may be used in any text file that is extracted from a source code control system. There is no formal link between the source code and the Tableau system, so users may include tokens in their source code and then register them with Tableau later.

By default a token has the form: `%%TOKEN%%` where `TOKEN` is any text without spaces. The token delimiter may be changed at any time to something other than `'%%'` (see Chapter 26 - System Configuration).

As an example, you may have some Java code that looks like this:

```
...
public static final String appVersion =
"5.16b";
public static final String appName    =
"Tableaux";
...
```

You can replace the code with tokens to have the values change with the environment:

```
...
public static final String appVersion =
"%%INTERNAL_SYSTEM_TAG%%";
public static final String appName    =
"%%APP_NAME%%";
...
```

These values are then parsed and replaced by Tableau before the code undergoes the build stage. In this case, `APP_NAME` could take on the value `'Tableaux-Dev'`, `'Tableaux-Test'` and `'Tableaux'` for a dev, test, prod environment set.

Token Bundles

Token bundles are used to group common tokens together for ease of management. Bundles have to actively be applied to components at deployment time. Therefore not all tokens in the system apply to all deployments; only those bundles you choose will be applied to a deployment.

You can create tokens with the same name in different bundles. When bundles are applied to a deployment there is an order of precedence that allows tokens to override lower token's values.

Bundle IDs

Every bundle has a unique bundle ID. This ID is assigned to each bundle as it is created. The IDs are mostly hidden from view and so users should not have much contact with them.

Creating a Bundle

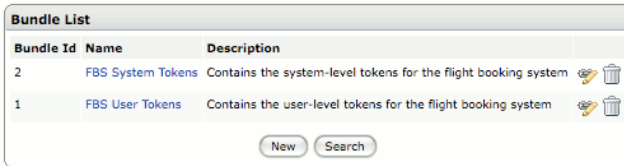
To create a token bundle, select **"View -> Token Bundles"** from the menu. You will be taken to a screen displaying the current bundles. Click on the **New** button at the bottom the screen.

The screenshot shows a dialog box titled "Token Bundle". It contains two text input fields: "Name:" and "Description:". Below these fields is a "Save" button.

Enter the name and a description and click on the Save button.

Modifying/Deleting a Token Bundle

From the menu, select "View -> Token Bundles". A list of token bundles is displayed.



Click on the Edit Bundle icon for the bundle you wish to modify or delete.

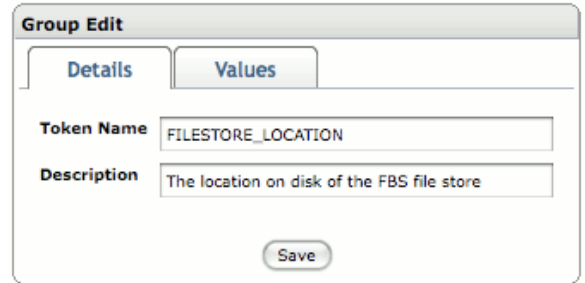
To modify the bundle, change the name and/or description and click on the Save button. To delete the bundle click on the Delete button.

Creating a Token

From the menu, select "View -> Token Bundles". Choose a bundle that will contain the new token and click on its name. This will display the tokens in the bundle.



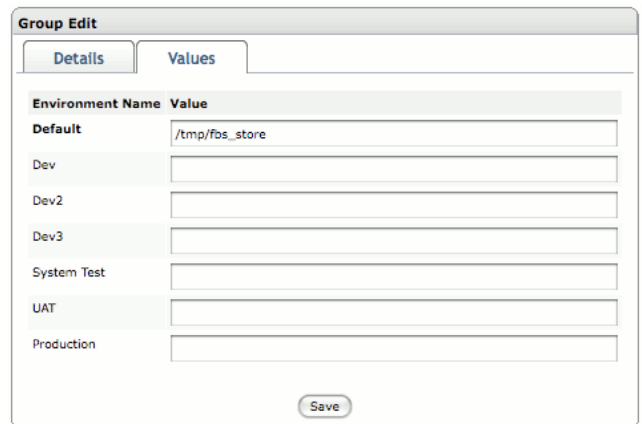
Click on the New button at the bottom of the screen. Enter the name of the token and a description and click on the Save button.



Note: The name of the token should be the string you place inside the %% bookends in your text file.

Modifying Token Values

Before a token is useful, you must fill in some values for each Environment that will replace it. Click on the name of the token, then choose the Values tab.

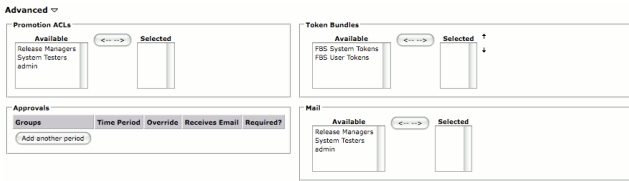


Fill in the token replacement values and click on the Save button at the bottom of the screen.

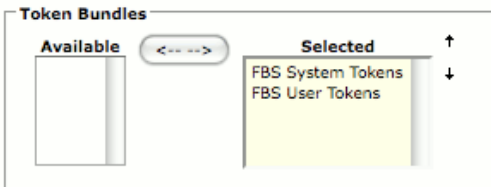
The "Default" value will take effect unless overridden by an environment-specific value.

Assigning

Before values from a Token contribute to a build, the Bundle containing the tokens must be "assigned" to the Component. To assign a Bundle to a Component, enter the Component Edit screen (either from the main work screen or from a Product screen) and choose either Default or a specific Environment. Click on the Advanced icon.



Select any or all of the token bundles to add. You may re-arrange the ordering of the bundles, keeping in mind that bundles higher up in the list override the values of tokens contained in bundles lower down in the list.



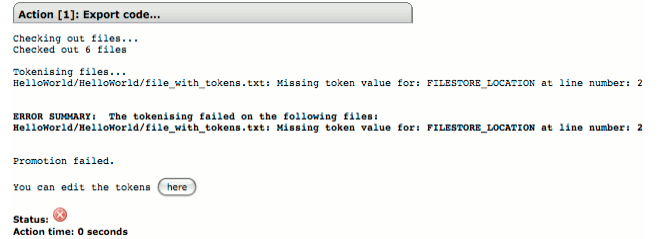
Click on the Save for the changes to take effect.

Missing Tokens

One important feature of the token system is that any text of the format '%<text>%' is

considered a token, whether there is actually a defined token in Tableaux or not.

If Tableaux encounters a token in a source text file, but no token has been assigned, then an error will occur during deployment and the deployment will fail.



If the token exists and has been assigned to the deployment, but the token value is blank (because you have not filled in a value yet) then this will *not* result in an error.

One method to determine if you have provided all the tokens the source code requires is to simply deploy the Component. The deployment will fail, but you will be informed of all the missing tokens.

A marginally cleaner method is to select the checkbox title "Don't promote, just search for tokens" in the extra Options box from the deployment screen. This option forces the deployment to only check the code out and conduct token searching and go no further.

Tokens Within Tokens

Sometimes you have may some token values that are made up of other tokens. For example you may have a token defining a base directory:

```
MY_BASE_DIR = /var/spool
```

You can use this value as the basis of other tokens:

```
MY_TEMP_DIR = %%MY_BASE_DIR%%/tmp
MY_DISTRO_DIR = %%MY_BASE_DIR%%/dist
```

This is perfectly acceptable in Tableaux. Keep in mind that tokens within tokens will also be replaced according to the order of precedence, not necessarily by tokens in the same bundle.

Standard System Tokens

There are some standard tokens defined by the system that anybody may use. These tokens are replaced with system values during deployment and are useful for providing information about the promotion itself. The standard tokens are:

- `INTERNAL_SYSTEM_TAG` - replaced by the tag/label of the deployment.
- `INTERNAL_SYSTEM_BUILD` - the build number of the deployment.
- `INTERNAL_SYSTEM_DATE` - the start date of the deployment.
- `INTERNAL_SYSTEM_USER` - the name of the user who initiated the deployment.
- `INTERNAL_SYSTEM_ENV` - the name of the destination environment of the deployment.
- `INTERNAL_SYSTEM_ENV_ID` - the ID of the destination environment of the deployment.
- `INTERNAL_SYSTEM_PLUGIN_TYPE` - the name of the Plugin used to conduct the deployment.

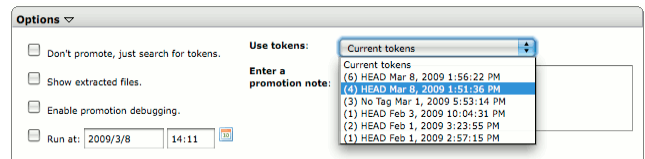
- `INTERNAL_SYSTEM_COMPONENT` - the name of the component being deployed.
- `INTERNAL_SYSTEM_COMPONENT_ID` - the ID of the component being deployed.
- `INTERNAL_SYSTEM_NOTE` - the contents of the note entered on the deployment screen.

Token History

The tokens used during every deployment are stored. From the main work screen or the Product screen, select the history button (📖) for a component. From the history screen you can choose the token history (↔).

History - Tokens	
Component:	HTML Files
Environment:	Development
Token Name	Token Value
FILESTORE_LOCATION	/tmp/dfs_store

You can also use the token history to re-create a deployment in the past by using the old token values for a deployment, rather than the current token values. In the deployment screen there is a drop-down box containing past deployments of the same component into the same environment. You may select the token set that was used in these deployments rather than the current set of tokens (as the tokens may have changed for the worse).



Token Templates

The working code on a developer's desktop cannot be compiled if it contains tokens. Therefore sometimes you want two copies of a file: one that developers use during development which contains hard coded values, and one that contains tokens.

Tableaux has the concept of token templates. Token templates are files that are identical to another file in the source code but that contain tokens instead of hard coded values.

During deployment Tableaux searches for token templates and replaces the non-tokenised file with the tokenised template.

A Tableaux template can be made for any file, and it has the same file name but with the extension: ".tblxtemplate".

For example, let us say a developer has a file in the source code named `src/numbers.java`. This file contains hard coded values so the file may be compiled on the desktop. However when deployed via Tableaux it will not be tokenised, so the developer creates another file called `src/numbers.java.tblxtemplate`.

When the code is deployed via Tableaux, the tokeniser replaces the tokens in the `.tblxtemplate` file and then moves it over the top of the original file: `src/numbers.java`.

Note: Remember that any changes made to the original file should also be made to the token template.

Files Not to be Tokenised

Sometimes you may not wish a file to be tokenised. An example may be when a file contains token markers but you don't want them to be parsed. This very document is one such example. This user manual contains '%%'s but when the user manual is generated via Tableaux, we do not wish them to be parsed.

To prevent a file from being tokenised simply create a companion file with the same name but with an extension: `.do_not_tokenise`.

Tokens in Component Parameters

You may also use tokens inside the component parameters. These tokens are replaced at deployment time, similar to tokens used inside source code files.

As a solid example, take the case where you are using CVS to provide source code. One component parameter is called "CVS Root". Instead of hard coding a host name in this field you may instead insert a token. Eg:

```
:ext:%%CVS_USER%%@%%CVS_HOST%%:/app
```

All these token values are then replaced at deployment time.

The order of precedence of competing tokens is also treated in the same manner as with normal files. Ie, the ordering of the token bundles containing identical token names defines the value the token is finally replaced with.

Part 4

Project Management

Goals.....	79
Goals.....	80
Creating a Goal.....	80
Editing a Goal.....	81
Organising Goals.....	81
Managing Access Permissions.....	81
Project Schedule.....	83
Viewing the Schedule.....	84
Creating an Activity.....	84
Editing an Activity.....	85
Public Activities.....	85
Reporting Graphs.....	86
Condensed Activity Graphs.....	87
Project Usage Graphs.....	87
Reporting.....	88
Generating a Report.....	89
Searching.....	90
Home Screen Search.....	91
Main Search Screen.....	91
Context Sensitive Search.....	91
OpenSearch Plugins.....	91

Goals

We've previously introduced you to Products, which are the main containers for a project. However, over time projects will typically undergo some iterations. The "Goal" is Tableau's method for organising these iterations.

Goals

Goals provide a means for segmenting phases of a project. Goals are designed to cover a program of work in terms of reporting, security and organising work effort, etc.

The details you see for a Product are actually the Goal's details. Every Product is created with a Goal called "-".

Creating a Goal

From the menu bar, choose "View -> Products". From the Products screen, choose the example product created earlier.

There is an area in the Product screen dedicated to the various goals of the product. At this stage, you'll notice that the drop-down has just a single entry "-".

Click on the New button.

Product - (edit)

Project Name: Flight Booking System

Description: The Flight Booking System is airline X's primary seat management system across it's entire fleet of aircraft.

Goal: V1.0

Created On: 3/8/09

Last Activity: 3/8/09

Notes:

Business Owner: John Smith

Created By: admin

Project Manager: (admin) Super User

Technical Manager:

Developers: (Scheduled) (admin) Super User

State: Active

Release Kit:

Environments: (0) Dev (1) System Test (2) UAT (3) Production (4) Dev2

Save Cancel

The next screen looks similar to the Product creation screen, however you now have the opportunity to enter the name of this Goal.

When initially creating the Product you were given the opportunity to enter specific details about it. When creating each subsequent Goal, you have another opportunity to set unique values. Each Goal within a Product can have its own project manager, developers, schedule and environments.

Each new Goal will initially inherit the values from the "-" Goal. However you may change any of these values at any stage.

Note: Only the default Goal ("-") allows you to override the default location that binary artefacts are stored (see Chapter 13 - Repository Plugins in the Internal Repository section).

Enter the name of the Goal: "V1.0" and click on Save.

You'll be back in the Product screen, but this time you can choose the Goal to display.



Editing a Goal

To edit either the default Goal (-) or any subsequent Goal, you must first ensure that you have that Goal selected.

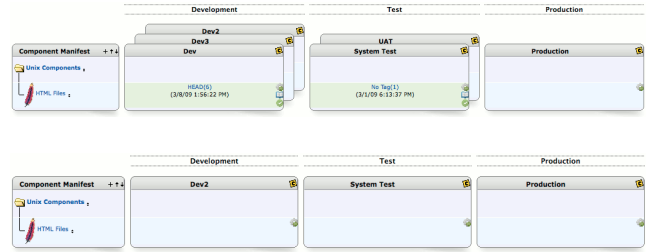
Click on the Edit button at the top of the Product box. You may modify any of the Goal's parameters from this screen. Click on the Save button when you are done, or click on Cancel to abandon your changes.

Organising Goals

The component manifest is common to all Goals. That is, if you create, modify, move or delete components then this change is reflected in all the Goals for the Product.

All other details are unique to the Goal. One important outcome from this is that each Goal can restrict environments to provide a different path-to-production than the others.

The two figures below are from the same product with two different Goals.



This is important for concurrent development to prevent clashing builds in development environments.

Another advantage of this segregation is that each Goal may have its project manager and set of developers. Thus, if required, separate teams can concentrate on their respective goals.

When deploying builds, each Goal will only display the history for that build, with one exception. The "-" Goal will display the entire history for each Component.

Managing Access Permissions

By default, the project manager, technical manager and developers all have access to the Product and Goal. You can change the particular access rights of each of these groups for each Goal.

From the Product screen, click on the Manage Permissions button.

Permission	Project Manager	Technical Manager	Developer
Edit Goal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
View Usage Graphs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
View Product Schedule	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Edit Product Schedule	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Manage Components	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Create Build	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Deploy Build	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Save Cancel

Only the product manager or Tableau administrators can modify the permissions for the Goal.

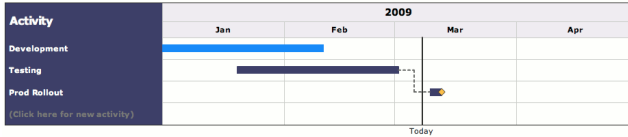
Project Schedule

Tableaux includes a rudimentary method for managing your project schedule. Your schedule ties heavily into staff and environment usage. Also, as Tableaux is the common meeting point for all staff on a project, it is also the ideal place to publish the working schedule.

The schedule also impacts on the work assigned to each user of Tableaux. As activities on the schedule are assigned to staff, it is added to the activities list on the main login page for each user. Thus, staff are always reminded of the tasks they are assigned to, especially across multiple projects.

Viewing the Schedule

From the Product screen, click on the Project Schedule button. This button is only available to managers and developers of the Product/Goal.



This screen shows the project activities on the left hand side, and the time period (as a gantt chart) on the right hand side.

The date range of the gantt chart is elastic and will change to accommodate the activities of the project.

The schedule will also display any public calendar entries.

Creating an Activity

To create an activity, click on the (Click here for new activity) text at the bottom of the chart. On the next screen you must enter the details for the activity.

At a minimum, enter a name for the activity, the start date and the duration. Click on the Save button.

Certain activities can be marked as:

- **Completed** - when the activity has been finished
- **Milestone** - when this activity achieves a mini-goal, such as completion of testing
- **Period** - when this activity should mark a certain period of the project, such as holidays.

There are three other optional fields:

- **Environment** - associates this activity with an Environment. This essentially books the environment for that period of time
- **Resources** - you can define the resources that will be working on the activity. This activity will appear in the My Activities list on their main portal
- **Depends On** - provides a dependency for this activity. For example, the UAT activity may depend on completion of a Unit Testing activity.

Editing an Activity

At any stage you can modify the details of an activity by clicking on its name in the gantt chart

This will take you to an edit screen that is identical to the Create screen.

Public Activities


There are some time periods that are common to all projects, for example major holidays.

To create or modify a public activity/period, from the menu bar, choose "**Administration -> Public Activity Calendar**".

Public Calendar			
Name	Start Date	End Date	Duration (days)
Public Holiday	2/5/09	2/6/09	1

New

You may create, modify or delete existing calendar entries from this screen. The public calendar entries are a cut-down version of the project schedule activities. You are only asked to enter the name and date/duration of the calendar item.

Edit Activity	
Name:	<input type="text" value="Public Holiday"/> *
Description:	<input type="text"/>
Environment:	<input type="text"/>
Start Date:	<input type="text" value="2/5/09"/> * 
Duration (days):	<input type="text" value="1"/> *
<input type="button" value="Save"/> <input type="button" value="Delete"/>	

The items that you create here will appear on all project's schedules.

Reporting Graphs

Tableaux provides an abundance of information to mine.

For interested parties, such as configuration managers, Tableaux provides condensed graphs of activities to provide an overall view of how various projects, environments or users are doing.

Also, each project can display details graphs on the health of the project. This includes deployment success/failure ratios, deployment statistics, and so on.

Condensed Activity Graphs

From the menu bar, choose "View -> Project Activities". The resulting screen lets you choose which activities you would like to graph:

- User by week
- User by month
- Total deployments into each environment and the success/failure ratio

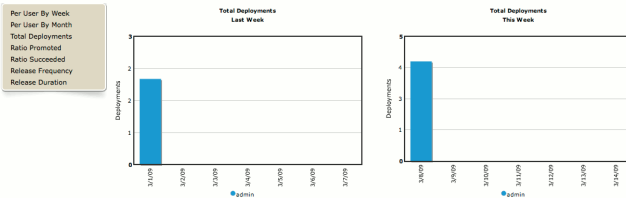
- The schedules of all products - This is a condensed graph showing one line per project/the timeline of the entire project
- Any conflicts of environment usage between projects
- Specific user schedules - this is the combined schedule of all project activities assigned to a particular user
- Specific environment schedules - this is the combined schedule of all project activities involving a particular environment.

All Products
All Conflicts
By User
By Environment



Project Usage Graphs

While in the main Product screen, click on the Usage Graphs button.



The subsequent screen allows you to break deployments down by:

Reporting

Tableaux includes a basic reporting tool that will generate deployment reports based on project, environment, user, date and other criteria.

These reports can be viewed online or downloaded as an Excel spreadsheet.

Generating a Report

To generate a deployment report, from the menu bar, choose "View -> Reports".

Generate Report

Environment

Product

Component

User

Tag/Version

Keyword

From / /

To / /

Return result as Spreadsheet:

Tableaux Report - 2009/3/7 - 2009/3/9									
Project ID	Project Name	Build #	Tag	Environment	Start Date	Time	User	Result	Keywords
2	HTML Files	4	HEAD	(0) Dev	Mar 8, 2009 1:51:35 PM	0s	admin	✓	
2	HTML Files	5	HEAD	(0) Dev	Mar 8, 2009 1:53:19 PM	0s	admin	✗	
2	HTML Files	6	HEAD	(0) Dev	Mar 8, 2009 1:56:21 PM	0s	admin	✓	
2	HTML Files	6	HEAD	(3) Production	Mar 8, 2009 4:44:04 PM	1s	admin	✓	RFC: 1234

There are several options available to customise the report. You may filter the results of the report based on:

- All environments or any particular environment
- All products or any sub tree of the component manifest.
- All users or any particular user
- Any specific tags or versions
- Any particular Keywords
- The start and end date of the reporting period.

Once you have selected the criteria of the report, click on the *Generate* button. The report will be displayed on the screen, unless you optionally choose to download the report as a spreadsheet.

Searching

As the number of Products and Components in Tableaux grow over time, it becomes more difficult to keep track of each one. Tableaux includes four methods to quickly find any objects within the system.

Firstly, Tableaux provides a search box on the home screen, from which you can search for any kind of object.

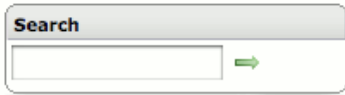
Secondly, Tableaux provides a main search screen from which you can search for specific types of objects.

Thirdly, many screens within Tableaux provide context sensitive search options for quick searching of relevant objects.

Finally, Tableaux also makes available search plugins for Firefox 2.0+ and IE7+. These plugins are an always-available field for easy navigation around the tool.

Home Screen Search

The home screen provides a search box from which you can search across all objects in the system.



This will return all objects (Products, Components, Release Kits, Tokens, etc) that match the search term.

Main Search Screen

To start searching for a specific kind of object, from the menu bar, choose "View -> Search". This brings you to a screen with multiple search options.

Search

ⓘ **Note:** You may enter multiple search criteria to narrow down your search.

Search for: Components ▾

Component ID:

Component Name contains:

Plugin: ▾

Product: ▾

Parameter containing:

Using Token Bundle: ▾

Deployed with tag:

Each search criteria is logically ANDed. Once you have selected a search option and entered the

search details, Tableau will locate all the objects that match your search criteria and present them to you

Component Name	Component ID	Project-Plugin	Product
HTML Files	2	Deploy HTML	Flight Booking System

Many of the results screens allow you to go directly to the object by selecting the hot-link associated with the object.

The search screen allows you to search for:

- Components
- Release Kits
- Tokens

Context Sensitive Search

Many screens provide search options directly on the screen. For example, you can:

- Search for component names on the main work screen
- Search for release kits on the release kit screen
- Search for tokens from the bundle screen
- Search for numerous objects from the main portal/home screen.

OpenSearch Plugins

OpenSearch is a standardised search plugin framework for Firefox 2.0+ and Internet Explorer 6+.

The search plugins are usually available on the top-right corner of the browser.

The main portal/home page of Tableaux provides contains instructions to the browsers on how to install the search plugins. For example, in Firefox if you click on the search drop-down you are given options to add the three Tableaux search plugins.

Once installed, you can simply type the name of the relevant object into the search box and Tableaux will conduct the search.

Please refer to your browser's documentation for more information on using search engines.

Part 5

System Management

User & Group Management.....	95
Creating a User.....	96
Adding Users to Groups.....	97
Modify/Delete Users.....	97
Creating a Group.....	97
Modify/Delete Groups.....	98
Changing Passwords.....	99
Per-User Customisations.....	99
Security Model.....	101
Security Rights.....	102
Best Practice.....	102
Object Attributes & Permissions.....	104
Viewing Object Attributes & Permissions.....	105
Changing Object Attributes & Permissions.....	105
Cascading Attributes/Permissions.....	106
Keywords.....	107
Creating a Keyword.....	108
Deleting a Keyword.....	108
Effects of Creating a Keyword.....	108
Querying by Keyword.....	109
System Configuration.....	110
Miscellaneous Items.....	111

Connection/Session.....	111
Directories.....	112
LDAP Authentication.....	112
Mail Options.....	112
Logging.....	113
Slave-Mode.....	113
Audit Logs.....	114
Viewing Audit Logs.....	115
Exporting Audit Logs.....	115
Digital Signatures.....	116
Enable Signatures.....	117
Signing Actions.....	117
Verifying Signatures.....	117
When to Enable Signatures.....	117
Deleted Object Repository (Graveyard).....	118
Viewing Deleted Objects.....	119
Restoring Deleted Objects.....	119
Licensed User Report.....	120
Report Setup.....	121
Report Options.....	121

User & Group Management

Tableaux is a multi-user system, so each user of the system has a separate user account. Permission to perform actions in the system are granted to user accounts by placing users into groups. Users can be assigned to more than one group, in which case the user receives the accumulation of security rights from all the groups.

Groups control access to the screens and functions in Tableaux. Groups can be organised into hierarchies which allows you to easily manage even large groups of users and their access levels.

This chapter details how to manage users and groups.

Creating a User

From the menu bar, choose "**Administration -> User Administration**". You will see a screen displaying the existing users in the system.

User Name	First Name	Last Name	Groups	Last Logged In
admin	Super	User	admin	January 3, 2010
u1234	John	Smith	admin	Never

New (Licensed User Report)

Click on the "New" button at the bottom of the screen.

User Edit

User Name:

First Name:

Last Name:

Email:

Enter Password:

Again:

Locked:

Enter the following details:

- Username - This is name by which the account is known.
- First Name & Last Name.
- Email - This is used by Tableau for various parts of its workflow.
- Password.
- Locked - Determines whether the account is active (unticked) or locked out (ticked).

The password field has a special relationship with the LDAP authentication system (see Chapter 3 - Before You Start). If the password field is filled, Tableau will *not* revert to LDAP for authentication. In other words, this password field over-rides any other form of authentication. This is especially useful for system accounts that do not have a presence in the directory server.

Click on the Save button to create the new account. The new account will have been created, but it will not have been assigned any security rights yet.

Auto-creating users

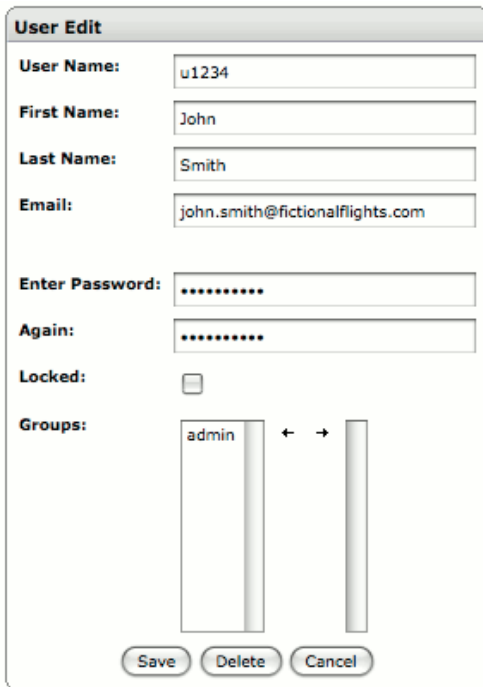
Under some special conditions it is possible to get Tableau to auto-create a user's profile when they first attempt to log in. This removes the need to manually create each and every user. Those conditions are:

- In the System Configuration screen (see Chapter 26, System Configuration) the Auto-create user profiles option is enabled
- The user does not already have an account in Tableau
- LDAP authentication to a Directory Server is enabled.
- The user has a presence in the Directory Server
- The user is in at least one group in the Directory Server that is linked to a group in Tableau.

Adding Users to Groups

To add a user to a group, you must first edit the user. Bring up the user list ("**Administration -> User Administration**") and click on the name of a user.

From the next screen you may choose the groups the user should belong to.



The "User Edit" dialog box contains the following fields and controls:

- User Name:** u1234
- First Name:** John
- Last Name:** Smith
- Email:** john.smith@fictionalfights.com
- Enter Password:** [masked]
- Again:** [masked]
- Locked:**
- Groups:** A list box containing "admin" with left and right arrow buttons next to it.
- Buttons:** Save, Delete, and Cancel.

You can also remove users from groups from this screen.

Click on Save for the changes to take effect. The user will see the effects immediately, even if they are currently logged in.

Modify/Delete Users

To modify or delete a user you must first enter the edit screen. Bring up the user list ("**Administration -> User Administration**") and click on the name of a user.

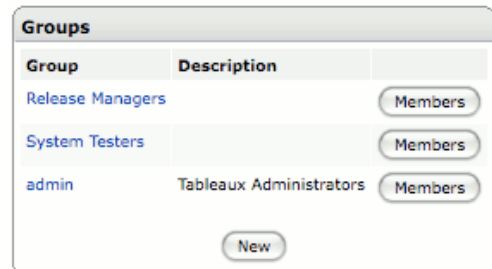
To modify the user's details, you can simply change any of the fields and click on the Save button.

To delete the user, click on the Delete button.

To cancel any changes, use the Cancel button.

Creating a Group

From the menu bar, choose "**Administration -> Group Administration**". You will see a screen displaying the existing groups in the system.

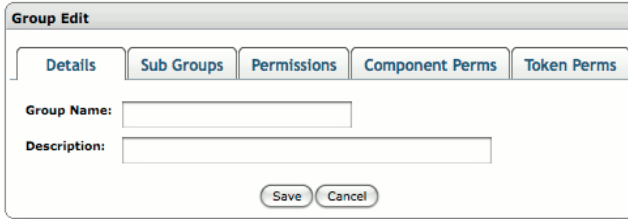


The "Groups" dialog box displays a table of existing groups and a "New" button at the bottom.

Group	Description	Members
Release Managers		Members
System Testers		Members
admin	Tableaux Administrators	Members

Buttons: Members (for each group), New (at the bottom).

Click on the "New" button at the bottom of the screen.



There are too many details to include in a single page, so the group's details are spread out over several tabs.

Details tab

- Group name - This is name by which the group is known.
- Description - A quick description of the function of the group.
- LDAP Group - If LDAP authentication is enabled (see Chapter 26 - System Configuration) then you can also specify a Directory Server group name that can also determine membership of the group.

Sub Groups tab

This tab allows group permissions to stack up in a hierarchy. By adding groups to the right-hand side of this screen, their permissions also apply to this group.

Permissions tab

This tab defines the outright permissions available to users in this group. This usually includes direct access to various screens within Tableau.

Component Perms tab

This tab contains a matrix of permissions. The matrix defines each of the Products and the access users in this group have to the components in the Product. For each product you can define:

- The ability to create, edit and delete components.
- The ability to edit the parameters for the components.
- The ability to deploy the components and introduce a new build into the system.
- For each of the Environment Pools, the ability to edit the parameters and deploy builds.

Token Perms tab

This tab also contains a matrix of permissions. This time it defines access to the token system. For each of the token bundles you can define:

- Access to see the tokens in the bundle.
- Access to modify the bundle (create tokens, etc).
- Access to view the token values for various Environment Pools.
- Access to edit the token values for various Environment Pools.

Click on the Save button to create the new group.

Modify/Delete Groups

To modify or delete a group you must first enter the edit screen. Bring up the user list ("**Administration** -> **Group Administration**") and click on the name of a group.

To modify the group's details, you can simply change any of the fields and click on the Save button.


To delete the group, click on the Delete button.

To cancel any changes, use the Cancel button.

Changing Passwords

All users can modify their password only if they are in a group that gives them that access. If your site uses LDAP authentication then it would be wise to remove access to this screen for all but the system administrators.

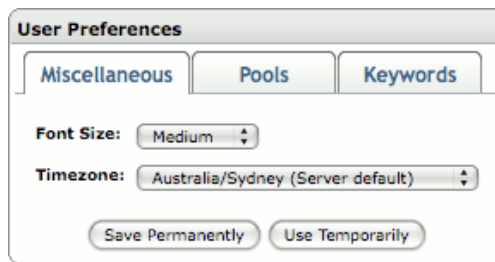
To change your password, choose "**Administration -> Change Password**" from the menu bar. Enter your existing password for validation, then your new password twice.



The system administrators may change the password for any user.

Per-User Customisations

Each user can customise aspects of how they interact with Tableau. From the menu, choose "**Administration -> User Preferences**".



In the first tab you may modify:

- Font Size - Alters the size of the text in the Tableau web interface.
- Timezone - Set the timezone that the user resides in (note: this does not change the timezone the server's timezone). Use this option when the user's timezone is different to the Tableau server's.
- Visible Environments - Voluntarily alters the environments that are visible to you. This is useful when the number of environments grows very large and you don't need to see them all.
- Foremost environment - Alters the Environment from each Pool that appears at the front of the stack in the main work screen.
- Keywords - Defines defaults for the Keywords for each Environment Pool.

The screenshot shows a 'User Preferences' dialog box with three tabs: 'Miscellaneous', 'Pools', and 'Keywords'. The 'Miscellaneous' tab is active. It is divided into three sections: 'Development', 'Test', and 'Production'. Each section has a 'Visible environments:' label with checkboxes and a 'Foremost environment:' label with a dropdown menu. At the bottom, there are two buttons: 'Save Permanently' and 'Use Temporarily'.

Section	Visible environments	Foremost environment
Development	<input checked="" type="checkbox"/> Dev, <input checked="" type="checkbox"/> Dev2, <input type="checkbox"/> Dev3	Dev
Test	<input checked="" type="checkbox"/> System Test, <input checked="" type="checkbox"/> UAT	System Test
Production	<input checked="" type="checkbox"/> Production	Production

You may save these changes permanently, or just use them for this session.

Security Model

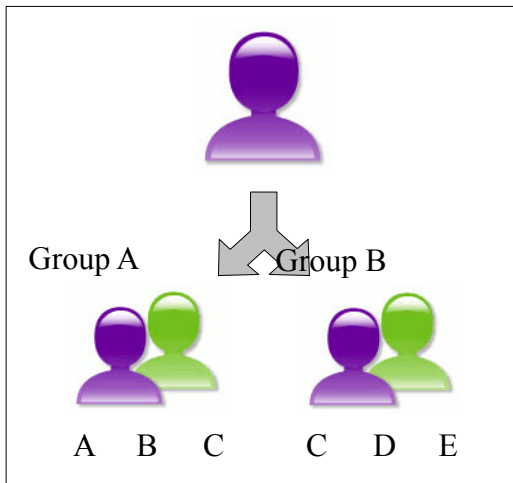
The previous chapter defined how to create Users and Groups. This chapter will aim to inform you of how placing Users into Groups, and defining particular security rights to the Groups affect what the Users can and cannot do in Tableaux.

Security Rights

A user's security rights are provided via one of two ways:

- By placing a user into a Group, the User gains all the security rights of the Group. A User's total list of rights is the sum total of all rights given to them by all the Groups they are in.

For example, in the diagram below, users in Group A are allowed to do A, B and C. Users in Group B are allowed to do C, D and E. So a User who is in both groups is allowed to do everything from A - E.



- Or by naming a user to a particular role for a Product. Each Product (and Goal) has 3 nominated roles:
 - Project Manager
 - Technical Manager
 - Developer

By naming a user into one of these roles, the user gets instant rights to parts of the Product. These rights are defined in the "Manage Permissions" button on the Product screen (see Chapter 17 - Goals).

Best Practice

There are many limitations to the level of access that can be granted via the product roles method. It is handy for a project manager to nominate the users on the project, but in terms of Tableau access it becomes quite complex to manage.

We recommend that most security rights be granted using user Groups. Work from a very coarse level down to a fine-grained level, rather than the other way around. If you start off micro-managing the groups then the overhead will make it too difficult to manage.

Remember that providing deployment access does not always mean that anybody can just deploy a component to a production environment, as the approval workflow system prevents any unauthorised deployments.

We therefore recommend that you organise your groups in the following way:

- Create a group called "Read-Only" that gives very basic access to all users. Security rights include:
 - Change Password
 - Reports,
 - Search
 - View Bundle List

- View Project-Plugins
- View Releases
- View Scheduler
- View Users
- Work Screen

Place all users in the Read-Only group.

- For each Environment Pool (Dev, Test and Prod) create:
 - A "Deployers" group - this group provides read-access to component parameters, and the ability to deploy components to any environment in the Pool, like this:

	Development			Test			Production		
	Manage components	Edit default params	Create builds	Edit params	Deploy builds	View history	Edit params	Deploy builds	View history
All	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Flight Booking System	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- A full Read-Write group - this group can modify component parameters and token values, as well include the rights from the Deployers group (the Sub Groups tab in the Group edit screen), like this:

	Development			Test			Production		
	Manage components	Edit default params	Create builds	Edit params	Deploy builds	View history	Edit params	Deploy builds	View history
All	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Flight Booking System	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Therefore you would end up with the following Groups:

- All - Development - Deploy
- All - Development - Edit
- All - Test - Deploy
- All - Test - Edit
- All - Production - Deploy
- All - Production - Edit

For each of the groups mentioned above, you should also include the rights from the Read-Only group using the Sub Groups tab in the Group edit screen.

Then, instead of micro-managing developers and creating scores of Groups for each project, simply place developers into one level or another for each Pool. Eg: "All - Development-Edit" and "All - Test - Deploy".

This Group layout caters for the many developers who often migrate between projects, or work in multiple projects concurrently.

- Create a full-access group for Infrastructure staff, as they have full access to the production environments anyway.
- Create Groups for Test teams, Release managers, etc as required.

Even for a large organisation, following the above practices will result in no more than a couple dozen groups.

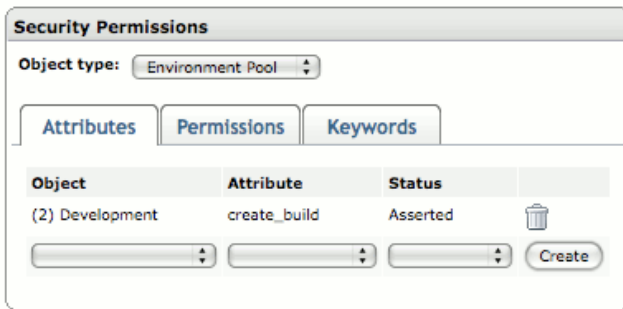
Object Attributes & Permissions

Many objects within Tableaux can have attributes and permissions set on them that affect how users can interact with them.

- Attributes are a single setting placed on an object that affects how that object works in Tableaux.
- Permissions are used to connect objects with Groups to define certain actions that are allowed on the object.

Viewing Object Attributes & Permissions

To view the current attributes, select **"Administration -> Permissions"** from the menu.



This screen looks very complicated, since it is used to define both attributes and permissions, but is easily broken down for ease of use.

Initially you must choose an object type. This can be one of either:

- Environment
- Environment Pool
- Product
- Bundle

For all object types, the attributes and permissions may either be asserted or revoked.

In general the attributes you can set on objects are:

- Locked - The environment or pool are locked from any further deployments.

- Create Build - Allows users to create new builds into the environment or pool (see Chapter 10 - Deploying). You must set this attribute on at least one environment or pool in Tableaux.

In general, the permissions you can set on objects are:

- Visible - Allows the object to be either seen or invisible to particular groups, depending on whether the permission is asserted or revoked.

Changing Object Attributes & Permissions

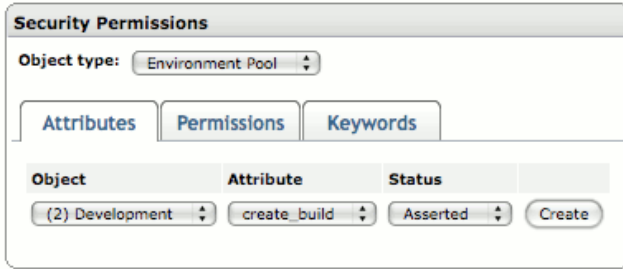
Open the attribute screen by selecting **"Administration -> Permissions"** from the menu.

Select the type of object you wish to set or change attributes or permissions for. For example, select Object type: Environment Pool.

Modify an Attribute

To modify an attribute, from each of the drop-down boxes select a particular object, an attribute, and a status. For example, allow users to introduce a new build into an environment, select "create_build" and "asserted" from the lists.

Click on the *Create* button to set the attribute.



To delete an attribute, simply click on the Delete button next to the attribute you wish to remove.

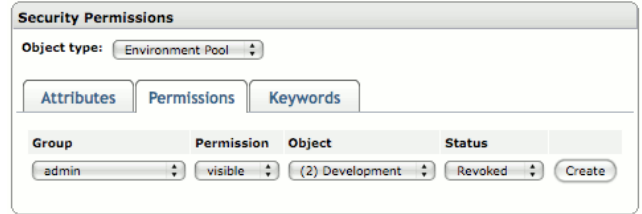
Note: You need to set the "Create build" attribute on at least *one* environment (either directly or via an environment pool), otherwise your users will be unable to deploy components. The most common case is to apply to the "Development" pool.

Modify a Permission

Modifying a permission is similar to an attribute, however you must select a group of users to apply the permission to. From the drop-down boxes you must select:

- A group of users the permission will apply to.
- The permission to apply.
- The object to apply the permission to. This list will only show the object types that you selected higher in the screen (Environments, Pools, etc).
- Whether to assert or revoke the permission. For example, to make an object invisible to the chosen group, you would "revoke" visibility.

Click on the Create button to set the permission.



To delete a previously set permission, simply click on the Delete button next to the permission you wish to remove.

Cascading Attributes/Permissions

One thing to keep in mind is that attributes and permissions cascade down to objects below the one with the attribute set.

The cascading rules are:

- Attributes and permissions set on Environment Pools cascade to all Environments in the pool.
- Attributes set on Products cascade to all Components in the Product.
- Attributes set on Bundles cascade to all Tokens within the Bundle.

Keywords

Keywords are a mechanism by which extra, site modifiable, fields can be captured during deployment of components. For example, if deployments into certain environments require an RFC number then you can create a "mandatory keyword" that will accompany each deployment, and becomes a field that users must fill in before the deployment can take place.

Creating a Keyword

In the previous Chapter we introduced the Permissions screen. Setting keywords is done on the same screen. From the menu bar, choose **"Administration -> Permissions"**. You will see a screen that allows you to modify various attributes of various objects.

In the Object Type box, choose either Environment Pool or Environment, depending on the granularity you require.

The screenshot shows the 'Security Permissions' dialog box with the 'Keywords' tab selected. At the top, 'Object type' is set to 'Environment Pool'. Below are three tabs: 'Attributes', 'Permissions', and 'Keywords'. A table lists existing keywords:

Object	Keyword	Status	
Test	RFC	OPTIONAL	
Production	RFC	MANDATORY	

At the bottom, there is a dropdown menu, a text input field, another dropdown menu, and a 'Create' button.

You can now enter a Keyword and apply it to the object you choose.

You must enter:

- The name of either an Environment Pool or Environment, depending on your selection earlier.
- The name of the Keyword, for example "RFC".
- Whether the field is Mandatory or Optional. For example, you might make an "RFC" Keyword optional in development environments, and mandatory in production ones.

Click on the Create button.

Deleting a Keyword

As above, choose **"Administration -> Permissions"** from the menu and select the type of object you'd like to delete the Keyword from (Environment Pool or Environment).

This displays the keywords on all Environments or Environment Pools.

Click on the Delete button to remove the Keyword field.

Effects of Creating a Keyword

When you create a Keyword, any deployments into an Environment with a Keyword applied now requires extra information to be entered.

After clicking on the Deployment button from the main work screen or a Product screen, you will notice an extra box.

The screenshot shows the 'Extra' dialog box with the 'Keywords' section. It contains a label '*RFC:' followed by a text input field.

The text entered into this field becomes attached to the deployment and appears in the deployment history.

Build	Tag	Start Date	Time	User	Result	Goal	Keywords	Actions
6	HEAD	Mar 8, 2009 4:44:04 PM	1s	admin	✓	-	RFC: 1234	

Build Compare Add History

Querying by Keyword

Keywords also become search candidates. In Chapter 20 - Reporting you were introduced to the reporting screen. From this screen you can report on particular values of the Keywords field.

System Configuration

The System Configuration set of screens allows you to modify how Tableaux runs. You've already been introduced to these screens in Chapter 3 - Before You Start.

The screens are broken down into five categories:

- *Miscellaneous Items* - Covers all items that don't fit into the other areas.
- *Connection/Session* - Controls connection and sessions timeouts, etc.
- *Directories* - Controls where Tableaux finds its various components in the file system.
- *LDAP Authentication* - Controls authentication to an LDAP-enabled directory server.
- *Mail Options* - Controls various mail workflow parameters.

Miscellaneous Items

From the menu, choose "Administration -> System Configuration". Select the "Miscellaneous" tab.

Setting	Value	Description
Token Separator:	%%	This value defines the delimiters that causes Tableaux to identify a string as a token in the source code.
Default Repository Plugin:	CVS	All new Components will be created with this repository plugin by default.
Default Binary Repository Plugin:	Main Repository	When creating a binary repository for Tableaux to snapshot binaries into, the repository will initially be created with this plugin.
Continue release on fail:	true	Continue a release even after encountering a failed sub project.
Re-tag on fail:	true	Place re-tag markers on source code even if the first promotion fails.
Message of the day:		
Read-Only mode:	false	Lock Tableaux and make it Read-Only
Cache size:	2000	Size of the internal cache. This uses more memory, but Tableaux operates faster.
Require digital signatures:	false	Digital signatures are a method of non-repudiation that provides an iron-clad audit trail. Enabling this feature will put a password field on all deployment screens that require approval.

Items in this section include:

- The token separator - This controls the "bookends" that define a token. The default value is "%%".
- Encrypt values in the database - This forces Tableaux to encrypt all sensitive information (passwords, etc) in the database.
- Default Repository - You should set this to the source code repository flavour that you use most often.
- The default binary repository - See Chapter 13 - Repository Plugins under the Internal Repository section.
- Debugging - This turns debugging on and off. Debugging will appear in Tomcat's stdout file (under the Logs directory). This controls internal debugging which includes repository plugin debugging. It does not affect debugging of deployment scripts in Project Plugins.

- Continue Release Upon Failure - Controls whether release kits continue deployment once they hit a component that fails.
- Re-tag Markers Upon Failure - Controls whether Tableaux re-tags files in the repository even if a deployment fails.
- Message Of The Day - This text snippet appears on the front page and is used to disseminate information to the users of Tableaux.
- Lock Tableaux - Setting this option puts Tableaux into a read-only mode and all deployments are disallowed.
- Internal Cache Size - If you have sufficient memory, you may increase the cache size to increase the speed of Tableaux by reducing the number of database accesses.
- Digital Signatures - Ask for a digital signature when performing sensitive actions, such as deploying or approving jobs.

Connection/Session

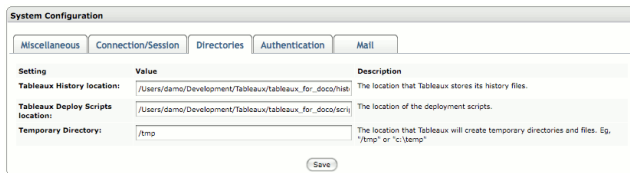
Setting	Value	Description
Max concurrent deployments:	10	The maximum number of jobs that can be promoted concurrently. This will be a function of the CPU power of the Tableaux server.
Number initial database connections:	2	Initial number of database connections. Database connections take a relatively long amount of time to establish. Creating more initially increases the startup time, but may save time later on a heavily loaded system.
Max database connections:	10	Enter the maximum number of database connections. This caps the number of connections which reduces memory overhead and load on the database server but may create delays for Tableaux users if all the connections are busy.
User session timeout:	60 minutes	When users close a browser running a Tableaux session, that session is left idle for a certain period of time. Over public networks this time should be reduced as much as is practical to reduce security risks.
Screen refresh interval:	30 seconds	This value defines the interval of the refresh in the Work screen. On a heavily loaded system increase this number to reduce load.
Check Sequence Id:	false	The Sequence ID is a token passed to the client for authentication purposes in order to prevent a replay attack on Tableaux. This greatly enhances security over unsecured networks (eg, the internet). However it is usually not required if Tableaux is only accessed over your LAN.
HTTP Proxy Host:		These 4 parameters allow Tableaux to connect to the internet through a proxy.
HTTP Proxy Port:		
HTTP Proxy Username:		
HTTP Proxy Password:		

Items in this section include:

- The size of the concurrent deployment queue. This number defines how many deployments TableauX can undertake at any one time.
- Initial database connections.
- Maximum database connections.
- Session Timeout - This controls how long user's sessions can idle before they are destroyed. Lower this figure for public networks, and raise it for private networks.
- Refresh Interval - Controls how often (in seconds) the main work screen is refreshed.
- Proxy Details - Define the host, port, username and password for an internet proxy. TableauX requires internet access to fetch Plugins. TableauX does *not* pass any information to Incanica other than the version of TableauX you are running (for Plugin compatibility).

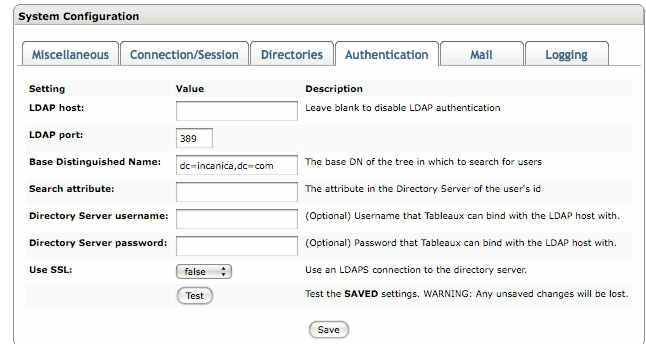
Directories

Directories are already covered in Chapter 3 - Before You Start.



LDAP Authentication

Authentication to LDAP-enable directory servers is already covered in Chapter 3 - Before You Start.



Mail Options

From the menu, choose "Administration -> System Configuration -> Mail Options".

System Configuration

Miscellaneous | Connection/Session | Directories | Authentication | Mail

This section defines the various mail workflow parameters. You may use any of the following tags in the fields below and they will be replaced at runtime with the stated values.

%st Start time of the promotion
 %ft Finish time of the promotion
 %dt Duration of the promotion
 %pid Project ID
 %pn Project Name
 %ei Environment ID
 %en Environment Name
 %us User who initiated the promotion
 %ta Tag/Version of the promotion
 %id Build number of the promotion
 %re Result of the promotion [SUCCESS|FAILURE|UNKNOWN]
 %rs The output from the promotion

Setting	Value	Description
Mail server:	localhost	A mail relay host that can deliver mail for us (usually "localhost").
"From" user name:	Tableaux	Username that TableauX mail appears to come from.
Subject for a successful promotion:		
Message body for a successful promotion:		
Subject for an unsuccessful promotion:		
Message body for an unsuccessful promotion:		

Save

Items in this section include:

- The server running an MTA. This is usually the local server. This server can deliver mail for us.
- The username that mail comes from.
- The subject and contents of an email sent as the result of a successful deployment.
- The subject and contents of an email sent as the result of a failed deployment.

Logging

The logging options allow you to alter the logging output of TableauX.

System Configuration

Miscellaneous | Connection/Session | Directories | Authentication | Mail | Logging | Slave Mode

Setting Value Description

Logging Level: FINEST Log all output at the indicated level. Note: the location of the log file can be set below.

Log file location: /var/log/tableaux/server/%p.log The log file. You can use %p as a placeholder for the incrementing rotation number.

Max size of log file: 5000000 bytes Maximum size a log file will reach before TableauX rotates it.

Max number of log files: 5 Number of log files to keep.

Save

You can change the logging level, the retention period, and the location of the log files.

Slave-Mode

The slave options allow you to put this TableauX instance into slave mode.

System Configuration

Miscellaneous | Connection/Session | Directories | Authentication | Mail | Logging | Slave Mode

Slave settings:

Setting Value Description

Slave mode: Check this to put this instance of TableauX into slave mode.

Listen Port: 8899 The port to listen on for connections from the master TableauX instance (ensure this is an unprivileged port: >1024).

Master's IP Address: (Optional) The master TableauX server's IP address that will be connecting to this slave. If set, an incoming connection from the master will be evaluated against this value, offering a (very mild) extra layer of security.

Master's Key: Copy and paste the Master's key into this field. This key is provided by the Master TableauX server once you have set up Master/Slave mode.

Save

When converting the TableauX instance into Slave mode, you must specifically tie it to a Master server instance. The Master Server will authenticate itself using a key. You must enter the Master server's companion key into the text field to have this TableauX trust the Master. For further information, see Chapter 14 - Master/Slave Mode.

Audit Logs

Every action in Tableaux that modifies the system causes an entry to be written in the Audit Log. Tableaux contains a read-only screen where users with the appropriate security rights may search the audit logs for particular actions. Tableaux records information about every action, including:

- Time and Date of the action.
- User who caused the action.
- The Environment/Product the action was performed on.
- A description of the action explaining what happened.

Viewing Audit Logs

Open the read-only audit screen by clicking on "Administration -> Information -> Audit Logs" from the menu.

Date	Component ID	Env ID	User	Description
12/12/10 7:33:16 PM			admin	Logged in
12/12/10 7:31:40 PM			admin	Logged in
12/12/10 7:30:39 PM			admin	Logged in
12/12/10 7:27:24 PM			admin	Logged in
12/12/10 5:13:51 PM			admin	Global Configuration changed
12/12/10 4:50:04 PM	2		admin	Saved component: 2
12/12/10 4:14:47 PM	2		admin	Saved component: 2
12/12/10 3:50:30 PM	5		admin	Saved component: 5
12/12/10 3:40:38 PM			admin	Logged in
12/12/10 3:42:18 PM			admin	Logged out
12/12/10 3:21:45 PM	5		admin	Created project: 5
12/12/10 3:21:28 PM			admin	Logged in
12/12/10 2:36:28 PM			Automatic	Database upgraded to: 5.4.4
12/12/10 2:36:28 PM			Automatic	Database upgraded to: 5.4.2
12/12/10 2:36:28 PM			Automatic	Database upgraded to: 5.4.1
17/11/10 3:17:48 PM			admin	Global Configuration changed
17/11/10 3:17:29 PM			admin	Logged in
12/09/10 6:06:15 PM	2	0	admin	Promoted HEAD
12/09/10 5:19:56 PM	2		admin	Saved component: 2
12/09/10 5:18:47 PM	2		admin	Saved component: 2
12/09/10 4:39:18 PM			admin	Logged in

Date from: dd/mm/yyyy Description: Env: Num rows: 100 (returned: 100) Search
 Date to: dd/mm/yyyy Comp ID: User: Export

From the audit log screen you can:

- Select a date range to display.
- Search for string in the Description field.
- Search changes to a particular component.
- Select all changes made to a specified environment.
- Choose all changes made by a particular user.
- Alter the number of lines displayed.

Exporting Audit Logs

Click on the **Export** button on the Audit Logs screen.

Export Audit Logs

Export Period: Past day

Export to directory: /usr/local/tableaux/export

Scheduled export:

Save
Export Now

The Export Period drop-down box allows you to select the amount of logs to send to the exported file. If you auto-schedule the job (as opposed to running a once-off export) then this will also coincide with the regularity of the export.

Choose a suitable location to export the logs to. This directory must be writeable by the user running the Tableaux server.

Ensure the Scheduled export checkbox is ticked to start the scheduled export. Or you can alternatively click on the Export Now button to perform a once-off export.

Digital Signatures

Tableaux provides a non-repudiation system via the use of digital signatures.

Sensitive deployments can be designated as requiring of a digital signature. When this occurs, the user is challenged for their password to complete the action.

Each user has a crypto key-pair which, combined with their password, is used to sign the job. These signatures are then stored in the database and/or onto a write-once medium.

Enable Signatures

To enable digital signatures, choose "Require Digital Signatures" on the Miscellaneous tab of the System Configuration (see Chapter 26-System Configuration).

Require digital signatures:

Signing Actions

With the signature option enabled, the following scenarios require a signature:

- Deploying a Component that requires approval.
- Deploying a Release Kit that requires approval.
- All approvals of all tasks.

Under all these scenarios, TableauX will prompt the user for their password before the deployment will proceed.

Signature Required

Verify password:

Verifying Signatures

To review and verify the signatures for a deployment task, you must click the (🔑) icon in the history screen.

A screen will appear showing each signature involved in the deployment.

Created	User	Message	Verified
Jun 13, 2010 5:29:44 PM	admin	Deploying component with details: Date: Sun Jun 13 17:29:42 EST 2010 Component ID: 2 Environment ID: 1 Dev tag: HEAD System tag: HEAD1236480695353 User initiating the job: admin	✔
Jun 13, 2010 5:30:13 PM	admin	Approving deploying of component with details: Date: Sun Jun 13 17:30:13 EST 2010 Type of approval: Approved Component ID: 2 Into environment ID: 1 Dev tag: HEAD System tag: HEAD1236480695353 User initiating the job: admin User approving/denying the job: admin	✔

The details of each signature is displayed, along with a verification icon. The icon is controlled by a cryptographic verification that neither the signature, nor the deployment job, have been tampered with.

If a signature is not successfully verified, then you know that either the signature or the deployment details were tampered with. In this event, you cannot rely on the data being accurate.

When to Enable Signatures

Digital signatures are essential for secure environments, such as Defence or corporate organisations, where it is important to verify that certain actions were authorised by appropriate people.

Digital signatures can be written to a secure, tamper-proof medium. Please contact Incanica Support if you have this requirement.

Deleted Object Repository (Graveyard)

When you delete a Component, Release Kit or User from Tableaux, the object is still kept in the database. It is possible for you to recover deleted objects by using the Graveyard screen.

Also, as Release Kits are modified, each iteration of the Kit is stored off. The Graveyard is the location where you will find these backups.

Viewing Deleted Objects

Open the graveyard screen by clicking on "View -> Graveyard" from the menu.

From this screen you can select the type of object. You can select from:

- Components
- Release Kits
- Users

Selecting an object type will display all deleted objects of that type.

Component Id	Component Name	Type
3	Database Component	Oracle8-Schema

Restoring Deleted Objects

To restore an object, click on the checkbox next to the object. You may select more than one object to restore at a time.

Restoring Components

To restore a Component, you must first select a product to restore to then click on the restore button. Your Components will now be visible in your chosen Product (see Chapter 7 - Creating Your First Product).

Restoring Release Kits

To restore a Release Kit, simply click on the restore button. Your Release Kit will be visible in the normal Release Kit screen (see Chapter 15 - Release Kits).

Restoring Users

To restore a User, simply select the user(s) you want to restore and click on the Restore button. The User will be visible in the normal User screen (see Chapter 22 - User & Group Management. Note: restoring a User will not return the User to any groups they were in before they were deleted.

Licensed User Report

Tableaux includes an inbuilt report to make it easy for you to keep track of your total licensed users.

Report Setup

Open the report screen by clicking on "Administration -> User Admin" from the menu. At the bottom of the screen, click on the Licensed User Report button.

Tableaux will display a screen like the following:

You have either run the user report right now, or schedule the report to run later.

If you auto-schedule the report, then you can choose to run it:

- Daily,
- Weekly,
- Monthly; or
- Yearly

If you auto-schedule the report then you can define who will receive the report by entering one or more email addresses, separated by new lines, in the Destination Address field.

Report Options

There are two customisable options for the report. They are:

- The group(s) to exclude from the report - if a user is *only* in one of the listed groups then they will be excluded from the report. The common scenario here is to specify the read-only groups.
- Days Idle - exclude users from the report if a user has not logged onto the system within the specified number of days.

For example, you can use these options to report on the number of users who have logged on in the last 30 days who is not a read-only user.

Part 6

Workflow

Approvals.....	123
Viewing Approvals.....	124
Setting Approvals.....	124
Approving Requests.....	125
Time Periods.....	125
Scheduler.....	126
Viewing Scheduled Jobs.....	127
Creating a Scheduled Job.....	127
Modifying/Deleting/Pausing a Scheduled Job.....	128
Command-Line.....	129
Running the CLI From the Command Line (or DOS prompt).....	130
Integrating the CLI into a Java Program.....	130
Troubleshooting.....	132
Installation problems.....	133
Performance Problems.....	133
General Problems.....	134

Approvals

Tableaux includes an approval engine, with the goal of being as unobtrusive as possible.

Certain objects such as Products, Components or whole Environments, can have deployment restrictions placed on them. With restrictions in place, any request to deploy is held in a bucket until approval is granted. If approval is granted then the deployment is run as normal.

Granting approval rights is most commonly conducted at the Environment level. For example, production environments would normally require approval from several groups of users. Whereas test environments would require fewer hoops.

Deployment restrictions can also be placed during certain time periods. Differing time periods can have different approval requirements. For example, one set of users may be set up for business hours approvals, while emergency approval can be granted to another set of users for out-of-hours deployments.

Viewing Approvals

Restrictions can be placed on several types of objects. These objects are:

- Components
- Groups of Components
- Environments

Component Level Approvals

Edit a component by clicking on its name in either the main work screen or from the Product screen.

Click on Page 2 to edit the second page of options.

The existing approvals are displayed in a dedicated box.

Component Group Level

Setting approvals on Component Groups is functionally the same as setting them on Projects. You can simply edit the Component Group and set the approvals in the same manner as described above

Environment Level

Click on the Environment name in either the main work screen or a Product screen. This results in a pop-up. Click on the Approvals button to be taken to the approval editing screen.

Setting Approvals

Use one of the methods described above to open the approval edit screen.

Power to approve deployments is granted at the Group level. For approval access you must therefore enter:

- A Group - indicating which set of users are allowed to approve deployments.
- A time period for which the approval applies. A group may be covered by multiple time periods, but you must create a separate entry for each one.
- Override - Determines whether the users in the specified group can override all the others. Ie, the approval of one user in such a group is sufficient to pass a deployment request. This is very useful for out-of-hours deployments where you may be granting an emergency approval list.
- Email Notification - determines if users in the specified group receive an email when a request is waiting for approval.

- Required - Whether this group is mandatory or optional. At least one user from each mandatory group must give their approval before a deployment can commence.

All approvals are cumulative. For example, if there is a blanket approval for an environment, but the Component Group also requires approval, then users from BOTH groups must give their approval for the deployment to proceed. Of course, if both happen to specify the same group, then only one approval is required for the promotion to proceed.

In the following screenshot, two people are required to give their approval during hours before this deployment can proceed. However, only one person is required after hours.

Approving Requests

When a user attempts to deploy a component that requires approval, an email is sent to all parties that can approve the deployment. The deployment job will appear in the Approval screen (click on **“View -> Approvals”** from the menu).

For each request, the number of approvals granted and the number still required are displayed.

It takes just a single user to deny a deployment request for it to be deleted from the list.

Once all approvals have been granted, the deployment is released and is either initiated immediately or is held over for forward scheduling.

Time Periods

The approval system is based on periods of time. Select **“Administration -> Time Periods”** from the menu to enter the screen from which you can view and edit the periods.

From this screen you can pre-define the hours that apply to the period, enter a name for the period, and click the Save button.

To edit a period, select it from the list, modify the hours boxes and click the Save button.

Once you have created a new period, it becomes immediately available to the approval system.

Scheduler

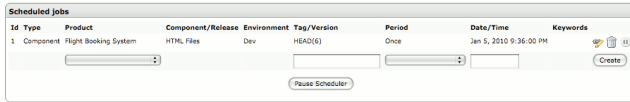
It is common for certain steps in a development methodology to revolve around automated, unattended builds. Such cases may occur during development where nightly builds are pushed to a test environment, or regular deployments into production during an unfriendly maintenance window prohibits system administrator involvement.

Tableaux provides a scheduling service to automate these and other tasks. It allows for nightly builds, cascading jobs, and one-off or periodic promotions.

The scheduler is fully integrated into the promotion request workflow to make it easy to set up scheduled jobs.

Viewing Scheduled Jobs

From the menu select "View->Scheduler". This will take you to the scheduler screen.



This screen displays all the currently scheduled jobs.

Each scheduled job contains the following fields:

- A scheduled job is either a release kit or a component.
- If it's a component then the Product that owns it is displayed
- The name of the product or release kit
- The name of the environment
- The tag or version of the component or release that will be deployed
- The scheduling period of the job. The periods are:
 - Once at a specified time in the future.
 - Recurring at a specified interval.
- The date/time the scheduled job will run
- Any Keywords attached to the job (see Chapter 25 - Keywords)

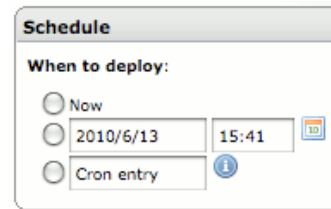
Creating a Scheduled Job

There are two methods to create a scheduled job:

- From the normal component deployment screen
- From the normal release deployment screen

Deployment Screen

If you choose any component and click on the Deploy button you are presented with several deployment options (see Chapter 10 - Deploying).



You can choose to run a job:

- **Now** - The job will not enter the scheduler and will be run immediately.
- **Once off** - Kick the deployment off at a specified time in the future.
- **Recurring** - Make this a recurring job. Enter a cron-style entry to determine when to run.

When entering a recurring deployment, you can enter a cron-style entry to determine when to run the job. The cron-style entry is composed of 5 numbers. These numbers are:

- **Minutes** - Minutes of the hour should the task run. Range from 0 - 59.
- **Hours** - Hour(s) of the day should the task run. Range from 0 to 23.
- **Days of Month** - Day(s) of the month should the task be run. Range from 1 to 31.
- **Months** - Months of the year should the task be run. Range from 1 (Jan) to 12 (Dec).
- **Days of Week** - Days of week should the task be run. Range from 0 (Sun) to 6 (Sat).

Some examples are:

- **5 * * * *** - Run on the 5th minute of every hour.
- *** 12 * * Mon** - Run on every minute during the 12th hour of Monday.
- **59 11 * * 1-5** - Run at 11:59am on Monday through Friday.
- **3-18/5 * * * *** - Run every minutes, starting from the 3rd minute of the hour to the 18th.

One important point to note is that a scheduled job will be deleted once it has completed. You do not need to manually delete a job once it has finished.

Release Screen

From the menu select "**Release -> Release Kit Promote**". This will take you to the release kit deployment screen (see Chapter 15 - Release Kits).

Similar to forward-scheduling a component deployment, there is an option on the Release Kit page to enter a date and time for the deployment to occur on.

Modifying/Deleting/Pausing a Scheduled Job

From the menu select "**View->Scheduler**". This will take you to the scheduler screen. Each scheduled job has three buttons available:

- Edit the job
- Delete the job
- Temporarily pause the job

Command-Line

The command line interface allows interaction with Tableaux beyond the web interface. The CLI is itself a Java application and so it is cross-platform.

The CLI allows you to link Tableaux actions into other 3rd party or bespoke programs, such as trouble ticket, environment management (QA), and HR applications.

The CLI can be operated in two modes:

- Run literally from the command line (or DOS prompt)
- Linked into another java application

There are two files that make up the CLI. They are:

- tableaux-cli.jar
- jargp.jar

Running the CLI From the Command Line (or DOS prompt)

To initiate the CLI, run the following command.

For Unix, run:

```
#> java -classpath jargp.jar:tableaux-
cli.jar \
    au.com.incanica.cli.CLI \
    -U "http://<server>:8080/tableaux" \
    -u <username> \
    -p <password> \
    -a <action> \
    [ extra parameters ]
```

For Windows, run:

```
c:\> java /classpath
jargp.jar;tableaux-cli.jar \
    au.com.incanica.cli.CLI \
    -U "http://<server>:8080/tableaux" \
    -u <username> \
    -p <password> \
    -a <action> \
    [ extra parameters ]
```

If you run the CLI without an action, you will be shown the complete list of actions. They are:

```
-a promotions.promote -P<component id>
-e<environment id>
-a promotions.result -P< component id>
-e<environment id>
-a approvals.list
-a approvals.approve -i<approval id>
-a approvals.deny -i<approval id>
-a pool.create -v "<pool name>"
-a pool.delete -z<pool id>
-a pool.list
-a environment.create -z<pool id> -v
"environment name"
-a environment.delete -e<environment Id>
-a environment.list
-a environment.list -z<pool id>
```

```
-a product.list
-a product.create -v "product name"
-a product.delete -S<set id>
-a bundle.list
-a bundle.create -v "bundle name"
-a bundle.delete -b<bundle Id>
-a token.list -b<bundle Id>
-a token.create -b<bundle Id> -v "token
name"
-a token.delete -T<token Id>
-a token.value.list -T<token Id>
-a token.value.fetch -T<token Id>
-e<environment Id>
-a token.value.fetch.tokenised -T<token
Id> -e<environment Id>
-a token.value.modify -T<token Id>
-e<environment Id> -v "value"
-a user.list
-a user.create -n "user name" -v "details"
-a user.delete -n "user name"
-a user.modify -n "user name" -v "details"
-a user.disable -n "user name"
-a user.enable -n "user name"
-a user.group.add -n "user name" -v
"group1, group2..."
-a user.group.remove -n "user name" -v
"group1, group2..."
-a group.list
-a group.create -n "group name" -d
"description"
-a group.delete -n "group name"
```

Integrating the CLI into a Java Program

You'll need to include both `jargp.jar` and `tableaux-cli.jar` in your classpath. To then make a call to `Tableaux`, use the following snippet:

```
import au.com.incanica.cli.Cli;
...
String commandToRun = "-U
http://...:8080/tableaux -u username -p
password -a environment.list";
try {
    String output =
Cli.execute(commandToRun.split(" "));
```

```
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    ...
```

An example of running the CLI is:

```
~> ./java -classpath  
jargp.jar:tableaux-cli.jar \\  
    au.com.incanica.cli.CLI \\  
    -U  
"http://localhost:8080/tableaux" \  
    -u admin -p xxxxxxxx \  
    -a pool.list
```

```
| Id | Name | Description |  
| 1 | Default | Default environment  
pool|  
| 2 | Development | Contains all the  
Dev envs|  
| 3 | Test | Contains all the Test  
envs|  
| 4 | Production | Contains all the  
Prod envs|
```

Troubleshooting

This appendix will help troubleshoot common problems with Tableaux. Please refer to this chapter to diagnose problems before calling Customer Support.

Installation problems

License file

If the main screen says that the product is not licensed then:

- Your license file may be corrupt or out of date.
- The license file may not be readable by the owner of the servlet engine.
- The license file may not be in the same location that is set in the deployment descriptor.

Check the Servlet deployment descriptor for the location of the license file. Ensure that the file exists in this location and is readable by the servlet engine. Ensure that the license file is not out of date by checking the expiry section.

Database

If the screen says that it cannot connect to the database then:

- The database server may not be running. If you are using the default database then check that the java process is running and that you can connect to it using the "sql" command.
- The deployment descriptor may have incorrect details. The hostname, username or password may be incorrect.

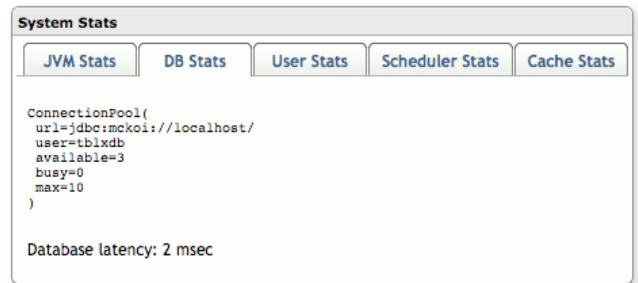
Performance Problems

Most performance problems stem from either a slow database or poor caching.

Select "**Administration** → **Information** → **Stats**" from the menu.

Database

Under the DB stats tab, check the database latency.



Ensure the latency is less than around 5 milliseconds. Any larger and Tableau will struggle to feel "snappy". If the latency is too high then you should look at your network, network interfaces, and the load on the database server.

Caching

The caching sub-system in Tableau exists to speed up the most commonly used objects in Tableau. If the cache sizes are too small then Tableau may be fetching more data from the database than it needs to.

You can check the cache sizes from the Cache Stats tab.

System Stats

JVM Stats DB Stats User Stats Scheduler Stats Cache Stats

Cache type	Max size	Current size	Age of oldest object	Avg hits/min	Avg misses/min	Hit rate	
Users	500	10	28 minutes	19.92	0.18	99.09%	Clear cache
Components	2000	36	44 minutes	7.58	0.60	92.67%	Clear cache
Environments	100	8	44 minutes	3.65	0.13	96.48%	Clear cache
Plugins	100	3	44 minutes	0.12	0.05	70.00%	Clear cache
Tokens	1000	0	-	0.00	0.00	0.00%	Clear cache
Release Kits	2000	9	29 minutes	2.37	0.18	92.81%	Clear cache
Security	1000	72	44 minutes	20.87	2.70	88.54%	Clear cache
Misc	100	3	28 minutes	4.17	0.05	98.81%	Clear cache
Resize cache		Clear all					

Tuning tip: For best performance, alter the Users and Release Kits max sizes to avoid them becoming 100% full.

In particular, size the User and Release Kit caches so that they are never full.

Increasing the size of the caches may provide very large performance increases, at the expense of requiring more memory (you can check memory utilisation from the JVM Stats tab).

General Problems

If you lose your administration password then please call Customer Support for help on resetting it.

Glossary

A _____

Action

A single script allowed by a Plugin to perform a particular task, such as compilation or deployment.

Approval

Part of Tableaux's workflow system, whereby requests for deployment into certain environments can be restricted until approval from appropriate people is granted.

B _____

Binary Repository

A location that Tableaux uses to store off binary files as a result of the deployment of a Component.

C _____

Component

The smallest unit that contributes to a project. For example, a .jar file, an .sql file, etc.

Component Manifest

The list of all components that make up a product.

Component Plugin

Provides the know-how on how to deal with specific digital artefacts. For example, how compile Java code or a .NET project.

D _____

Deployment

The act of applying a Component to an Environment.

E _____

Environment

A discrete level in a development life-cycle.

Environment Pool

A container that holds Environments for administrative purposes. Also holds Environments that have attributes in common. Eg, "Testing" pool, "Production" pool, etc.

F _____

First-level Environment

Any environment that allows builds to be introduced into it is a first-level environment. This applies to most Development environments, where as most Test and Production environments should only accept builds that have already been introduced into the system.

G _____

Goal

A Goal marks a phase of a product. Can be used to allow concurrent development on a product or simply to separate work chronologically.

K _____

Keyword

A short, user-enterable string that gets attached to a deployment. Used to attach customisable, searchable meta-data to a deployment.

P _____

Plugin

See "Component Plugin"

Product

A collection of digital artefacts that together make up a working project.

Project

A group of people working together on related components.

Promotion

Migrating a component or release from one environment to another. See also Deployment.

Promoting

See "Promotion".

R _____

Repository

A source of artefacts. This is typically a version control system. A version control system typically contains multiple repositories.

Repository Plugin

A plugin that allows Tableaux to talk to a particular repository, or version control system.

Release

Either the act of deploying one or several components (ie, to "release" a product into an environment), or a collection of components that are deployed as one atomic unit (see Release Kit).

Release Kit

A grouping of components, combined with possible manual tasks that describe the complete deployment, or part there-of, of a product.

S _____

Scenario

A named sequence of Actions to run at deployment time.

SCM

Short for Software Configuration Management. Generally a piece of software the allows version control of source code.

T _____

Token

A placeholder that earmarks a location that takes on a different value for each environment.

Token Bundle

An administrative collection used to group tokens into related areas/projects.

Index

Action.....	30	MySQL.....	9
Backups.....	17	Parameters.....	35
Component.....	27	Plugin.....	
Directories.....	14	Component.....	30
Environment.....	21, 24	Parameters.....	31
Environment Pool.....	24	PostgreSQL.....	9, 17
Goal.....	21	Product.....	20, 27
Group.....		Repository Plugins.....	57
Best Practice.....	102	Scenario.....	34
Creating.....	97	Scheduler.....	126
Deleting.....	98	System Requirements.....	7
Installing Tableaux.....	7	Tutorial.....	12
LDAP.....	14	User.....	
Licensing Tableaux.....	8	Creating.....	96
McKoi.....	17	Customising.....	99
Menu Bar.....	20	Deleting.....	97