Introduction to uRelease®

Concepts and Quick Start

Introduction to uRelease: Concepts and Quick Start

Publication date April 2013

uRelease Version: 4.1.2

Document Number: 4.1.2.2

About

This book describes how to use UrbanCode's uRelease product and is intended for all users.

This book is available in PDF and HTML formats at UrbanCode's Documentation portal: http://docs.urbancode.com/. uRelease's online Help is installed along with the product software and can be accessed from the product's web-based user interface. A PDF version is also included with the product's installation package.

Product Support

The UrbanCode Support portal, http://support.urbancode.com/, provides information that can address any of your questions about the product. The portal enables you to:

- review product FAQs
- download patches
- view release notes that contain last-minute product information
- review product availability and compatibility information
- access white papers and product demonstrations

Document Conventions

This book uses the following special conventions:.

- `Program listings, code fragments, and literal examples are presented in this typeface.`
- Product navigation instructions are provided like this:

  `Home > Components > [selected component] > Versions > [selected version] > Add a Status [button]`

  This example, which explains how to add a status to a component version, means: from the uRelease home page click the Components tab (which displays the Components pane); select a component (which displays a pane with information for the selected component); click the Versions tab (which displays a pane with information about the selected version); and click the Add a Status button.
- User interface objects, such as field and button names, are displayed with initial Capital Letters.
- Variable text in path names or user interface objects is displayed in *italic* text.
- Information you are supposed to enter is `displayed in this format`.

uRelease Overview

Today's fast-changing market challenges companies to release software on time and within budget. The rapid pace of innovation and technological change means today's plans are often out of date as soon as they're created. Against this backdrop discipline-centric teams, often geographically remote from one another, struggle to move applications into production. Automated tools by themselves cannot meet these challenges.

uRelease enables you to lay the track between pre-production and production and reliably run releases down the track. The *release train* can be provisioned by any type of rolling stock (automated, manual and ad hoc processes), and carry any type of freight. The release train's predictable schedule drives the release process. With uRelease you can integrate and synchronize team-based planning to arrive at a clear, open, and transparent plan. All stakeholders know the schedule and key milestones, and can be assured that the releases will depart on schedule and arrive on time.

Of course, as with any process occurring over time, unforeseen events happen. With uRelease you can identify problems as soon as they occur, accommodate unexpected events, and determine which features can be deferred until the next train leaves. uRelease's real-time feedback means plans can be refined and re-scoped to handle unforeseen issues as they arise.

uRelease's Web-based front-end ensures that the real-time collaboration between teams—the key to successful releases—is easy and intuitive, and makes long brutal releases a thing of the past. uRelease provides complete visibility into every facet of the software release life cycle, which is essential for predictable, and stable software deliveries.

uRelease can coordinate multiple deployments: those running concurrently with one another; those dependent on the output or status of other deployments—any combination imaginable. Dependences can be ordered and coordinated with middleware and networks. uRelease enables you to leverage current processes by interleaving manual tasks with automated process, and provides out-of-the-box integration with tools like uDeploy and uBuild.

Capturing the differences between release artifacts and processes is the basis for auditing and compliance reporting. uRelease's best-of-breed tools capture and log all activities throughout the release cycle, from pre-production through production. Key metrics are available for: compliance, governance, data aggregation and reporting, auditing/logging, and version/change control—in short, you can determine project status at any point and measure every aspect of the release cycle.

A partial list of uRelease's features include:

- real-time collaboration unites release teams with centralized task coordination and execution
- Web-based front-end provides comprehensive picture of current, past, and future releases
- template-driven release design
- extensive and open plug-in system provides integration with most popular third-party tools
- flexible team- and role-based security system
- out-of-the-box integration with uDeploy and uBuild

Concepts and Important Terms

uRelease uses standard system development life cycle and project management methodologies and practices. The following list some important terms:

- **release** A release is a repeatable plan used to drive deployments; it is uRelease's core mechanic. A release defines the "what, when, and how" of a release—it identifies the applications that will be released;

specifies when release events will occur; and, finally, defines tasks and the order in which they will be performed. A release can occur at any point in the development-operations life cycle, and can represent a major event in the life of a company, or be a comparatively minor event, like a recurring maintenance release. .

- **release environment** A release environment is a coherent logical unit into which software can be deployed. A release environment is composed of application-specific environments.
- **application** An application, as used here, refers to any business-meaningful piece of a system that can be independently deployed. Applications from all third-party integrations, such as uDeploy, are available for use. You can add any number of applications and applications from more than one third-party can be combined into a single release.
- **scheduled deployment** A deployment is the activity of putting some piece of software into a release environment.
- **deployment tasks** A task is a unit of work that can represent any business-meaningful activity associated with a release. Tasks can be added manually to scheduled deployments or be imported through CSV files.

Putting uRelease to Work

In general, you use uRelease at several key points during a release cycle:

1. Setup.
2. Plan a release.
3. Build, integrate, and test the system to be released.
4. Plan, practice, and execute production deployment.

The following sections summarize the activities performed to implement a release.

Setup Activities

Table 1. Setup Activities

Activity	Description
1. Installation.	Install uRelease as a Tomcat web application. See *Installing and Integrating*.
2. Configure integrations.	Make external objects available by configuring integrations. uDeploy applications and components, for example, become available once uDeploy is integrated with uRelease.
3. Define release environments.	Create the environments that will be mapped to release phases. When a release is created, you assign an environment to each phase.

Planning the Release

Start by determining the scope of the anchor functionality and assign a date that accommodates it, then work backwards from there. View each release as a date-driven project.

Table 2. Planning Activities

Activity	Description
1. Create or name the release.	Give the release a meaningful name and description; determine if it will be a major or minor release.
2. Applications.	Associate applications with the release.
3. Define path to production.	A release life cycle specifies the progression of environments through which software passes on its way to production. The life cycle does not indicate which specific environments are used for a release, but the general pattern, such as, DEV, INT, QA, UAT, PROD. It can also define the quality steps the software must successfully complete before being permitted to progress to the next environment. Finally, selecting a deployment plan determines how much orchestration and coordination is required for a successful deployment at given phase of the life cycle.
4. Identify the production date and known pre-production dates.	Known production and pre-production dates can be recorded and disseminated by scheduling deployments to the environments allocated to the release.
5. Define recurring windows.	Recurring windows can be used for deployments occurring on some regular interval, such as daily or weekly.
6. Define milestones.	Milestones are release-level checklist items that are tracked by date and status. See the section called “Define Milestones”.
7. Configure the release team.	Select a team to manage the release.
8. Add approvals.	An approval is a mechanism used to limit deployments to an environment regardless of quality (status) considerations, in order to ensure any work being done there is not interrupted.

Integrate and Test

A deployment can contain all applications in a release, a subset, or represent a one-off emergency deployment.

Table 3. Deployment Activities

Activity	Description
1. Schedule ad hoc deployments, if needed.	Ad hoc deployments can be scheduled at anytime so an exhaustive list of scheduled deployments is not necessary at the outset. Recurring windows can be defined and tested.
2. Update scheduled deployments.	Add specific application versions to deploy.

Activity	Description
3. Review the deployment plan's tasks.	Change tasks as required. Tasks can be added manually to a scheduled deployment, or can be imported through CSV files. Once a task is defined and saved, it becomes part of the deployment plan and is available for future deployments.
4. Certify application versions by applying quality statuses.	Quality statuses indicate that versions have met quality requirements. Statuses can be assigned manually, or through integration with external tools.
5. Grant gate exemptions.	Approvals and gates can be suspended whenever an emergency deployment is required.
6. Approve deployments.	An approval is a mechanism used to limit deployments to an environment regardless of quality (status) considerations.
7. Execute deployments.	Deployments are performed by running tasks defined in the deployment plan.
8. Complete milestones.	Update milestone status upon completion. Milestones extra-deployment items that can represent anything related to the release.

Execute Production Deployments

Table 4. Production Activities

Activity	Description
1. Verify deployment plan prerequisites.	A deployment plan consists of *segments*, which are groups of tasks that are to be completed at the same time. A segment cannot be started until all its prerequisites are completed. All segments can have prerequisites except the first one.
2. Verify overall schedule.	Ensure that all tasks have the expected duration; segment length is calculated using task durations.
3. Assign or claim tasks.	Tasks can be assigned to a role or to a specific user. Unassigned tasks can be claimed by anyone with the defined role.
4. Configure notifications	Notifications can be attached to plan, segment, or task, and set to trigger in several ways. Email notifications can be sent to users whenever user-defined trigger events occur.
5. Monitor deployments.	The dashboard provides a centralized portal into your releases, enabling you to obtain real-time statuses for an ongoing release from a single place.

Quick Start

This section provides information that will help you quickly become productive with uRelease. The following topics should be reviewed in order.

Planning the Release

Planning the release means answering some basic questions about its scope. Is it an entirely new release? Or does it use a previously defined plan? Perhaps it's a minor release—such as a patch—that requires almost no changes to an existing release? The answers to these questions will determine your path to production, and whether you can reuse an existing release, and if so, to what degree.

Tip

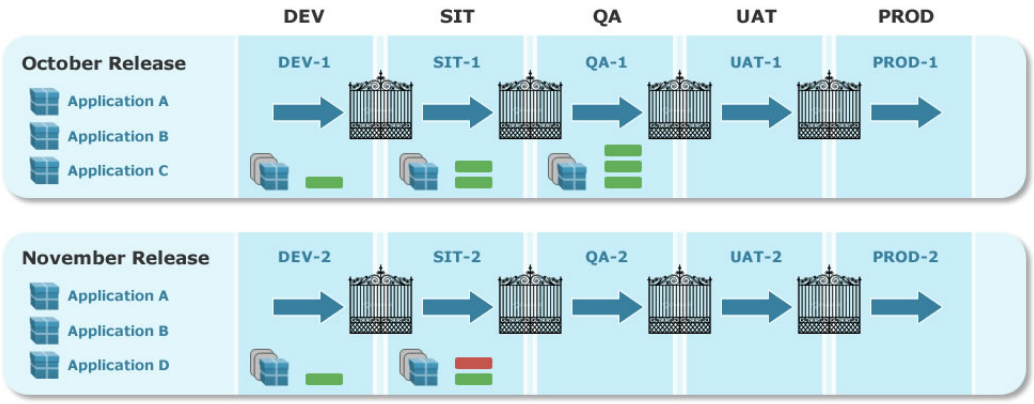
Ensure the inputs to the release train come from synchronized and open team-based planning. The goal is to define a set of clearly articulated deliverables and interdependencies.

The path to production refers to a succession of phases culminating in the final phase—production. At its simplest, a phase represents one or more environments and quality requirements. A phase can have additional items as well, such as quality statuses and gates.

The progression of phases is defined by a lifecycle model. When you create a release, the phases available to it are defined in the lifecycle model selected for the release. If a phase you need is not defined in a lifecycle, you can modify the model or create an entirely new one. uRelease provides a default lifecycle that you can modify as you see fit.

The following figure illustrates two releases—October, and November—that use the same lifecycle model. The phases defined in the model are listed across the top. Environments are allocated to releases and each phase is assigned one, which is shown in the illustration. The October Release, for example, uses the DEV-1 environment during the DEV phase, while the November Release uses DEV-2 for the that phase. The gates between phases are defined in the model.

Figure 1. Release Phases



It should be clear from the graphic that a lifecycle can be used for any number of releases. By varying the environments and applications (note that the lineup of applications differs between releases) you can craft releases for nearly any eventuality from the same lifecycle. And if a lifecycle is unsuitable for a

particular release—either too many phases or not enough, for instance—you can create a new lifecycle model at any time.

Create the Release

Narrowly speaking, creating a release means using the web user interface to give it a name, and select a lifecycle and team. More generally, it means determining if it is a major or minor release. As a rule-of-thumb, a minor release is one that can use an existing release's environments and applications or subset of its applications; a major release requires new environments and applications.

Associate Applications With the Release

Although applications are not required (you might create a release composed entirely of milestones and infrastructure-related tasks, for example), the majority of releases—by far—involve deploying applications. Applications can come from integration with external tools such as uDeploy, or be created within uRelease itself. Each release has all the applications defined in uRelease available to it. You can associate any number applications with a release.

For information about integrating uRelease with external tools, see the section called “Integrations”.

Define Path to Production

As discussed above, the phases available to a release are defined in the lifecycle selected for it. It might be helpful to think of a lifecycle model as a template used to create and drive releases. A lifecycle defines the progression of phases through which software passes on its way to production, which is represented by a production phase, or some similarly designated final phase. The lifecycle does not specify which particular environments are used for a release, but the general pattern. For example, a lifecycle might have the following phases: Development, Quality Assurance, and Production. Releases based on this lifecycle will have all three phases, although the actual environments used might vary from release to release. A lifecycle can also define the quality steps—gates—required to be successfully completed before software is permitted to progress to the next phase.

Tip

Develop strategies appropriate for each phase; strategies for high-ceremony production deployments might be inappropriate for low-ceremony environments.

The first step to define the path to production is to determine if an existing lifecycle model can be used, or if a entirely new one is required. Starting fresh with uRelease will naturally require the creation of lifecycles that mirror your normal processes and environments. As your experience grows, you will develop a stable of lifecycles that can handle most, if not all, of your releases. So the activities required to define the path to production will largely be determined by the availability of suitable lifecycles. The following tables outline activities based or whether or a not a reusable lifecycle is available.

Table 5. Creating a Lifecycle

Activity	Description
1. Name the lifecycle's phases.	Environments are mapped to phases.
2. Define Statuses.	Statuses are user-defined values, such as "Passed" or "Archived."
3. Add gates.	A gate is a mechanism that ensures items cannot be deployed into a phase/environment unless

Activity	Description
	they have a gate-specified status. Gates establish minimum entrance requirements to a phase.

Table 6. Using a Lifecycle

Activity	Description
1. Allocate environments to phases.	Identify the environments to be used during each lifecycle phase. A release environment is a user-defined construct that represents deployment targets; a release environment aggregates deployment environments.
2. Define approvals.	An approval is a mechanism used to control environments, regardless of quality (status) considerations. Approvers are designated by role; any user with the specified role can provide approval.
3. Select deployment plan.	A deployment plan defines how much orchestration and coordination is required for a successful deployment for a given phase.

Production Dates and Known Pre-production Dates

Known production and pre-production dates can be recorded and disseminated by scheduling deployments to the release environments.

Dates are defined when a new deployment is scheduled (*`Home > Scheduled Deployments`*).

Recurring Windows

A recurring window, or recurring deployment, are created for deployments that recur at predictable intervals. Recurring windows can be triggered hourly, daily, weekly, or by some *`cron`* expression.

Define Milestones

Milestones are activities—typically process checklist items—that must be completed in order for the release to remain on track. A milestone can represent anything that requires tracking, such as a kick-off meeting, operating system upgrade, or hardware and network upgrades. Milestones are tracked by date and status.

Milestones are attached to the release itself and not a particular phase or environment. Milestones are created on the Release page (*`Home > Releases > selected_release`*).

Tip

Although it might be tempting to concentrate on functionality identified during planning, remain alert to potential changes that re-scope or otherwise effect the release train's schedule.

Define Release Teams

Release teams, as you might guess, manage releases. A team consists of roles and users configured in the uRelease security system. A release must have at least one role configured for it. Typical roles are Admin, Release Manager, and User; all of which are available by default. Roles can be reused.

Roles are defined on the Roles page (*Home > Manage Security > Roles*).

Add approvals

An approval is a mechanism used to control environments, regardless of quality (status) considerations. A release that requires approval cannot proceed with a phase until approval is granted. Approvals are attached to phases. Approvers are designated by role; any user with the specified role can approve.

Integrate and Test

Once a release plan is finalized, it's time to test it and deploy it to non-production environments. Releases are implemented by *deployments*. A deployment targets a single phase and its associated environment (remember that each phase has a single environment associated with it). A deployment can be broad-based and use all applications in a release, represent a 1-off emergency deployment of, say, a single application, or anything in between. Deployments can be as precisely targeted as necessary.

uRelease's deployments brings together:

- the schedule that defines when the deployment will occur and specifies whether it is a one-time or a recurring event
- email notifications that are triggered by user-defined events and sent to a particular user or a user role
- required approvals

Deployment, or deployment plans, are composed of *segments*. A segment represents release activities intended to be completed together. A segment can be configured to run after the successful completion of another segment, or can run independently of any other segment. A deployment plan can have any number of segments. The default plan has two segments: Pre-Deployment Tasks, and Deployment Tasks.

Once a deployment plan is defined, a deployment can be initiated at any time with a *deployment request*. A deployment request can start an entire deployment or a portion of the plan, such as an individual segment.

Tip

Ensure every team has a fall-back plan in addition to its primary one. The fallback plan can be a simple as rolling-back to an old version until blocks are resolved.

Schedule Ad Hoc Deployments (Optional)

An ad hoc deployment is—as the name implies—an unplanned deployment. Ad hoc deployments can be scheduled at anytime, which means you do not have to define an exhaustive list of deployments during release planning.

Tip

Testing in typical environment progressions is important—including recurring winds—but be flexible enough to re-purpose environments if expected environments become unavailable.

Update Scheduled Deployments

Typically, a release's lineup of applications is defined when the release is created. Applications associated with the release are automatically available to any deployment using the release. Applications and

applications suites can be promoted to *released versions*. Typically, a released version represents an application (or suite) that has been successfully deployed and can reliably be reused.

Additionally, you can add applications to a release after deployments have been scheduled for it. New applications become part of any upcoming or in-progress deployment.

Define Deployment Tasks

Deployment activities are defined by *tasks*. An individual task is a unit of work that can represent any business-meaningful activity associated with a release. Tasks can be configured to run just once, or every time the deployment plan is used. A task can be assigned to a user role or specific user; and if unassigned, can be claimed by anyone with the requisite role. Once a task is defined, it is added to the task library and becomes available for other deployments.

When a task is created, it is given a *duration*, which is an estimate of the time it will take to complete. uRelease aggregates task durations to calculate overall deployment times.

Tasks can be automated or manual. Automated tasks come from integrations with external tools. Application processes from uDeploy applications, for example, are available as automated tasks in uRelease.

Manual tasks can represent any non-automated task, such as stopping or starting a server. Unlike milestones which are defined for the overall release, manual tasks (as well as automated tasks) are attached to a particular phase and segment. A segment can be considered a grouping of tasks intended to finish at the same time.

Typically, tasks are defined on the Scheduled Deployments page in the web application, but they can also be exported and imported (as CSV files).

Certify Application Versions

Application versions can have quality statuses. Quality statuses ensure that application versions have met some expected quality requirement. Statuses can be assigned manually, or through integration with external tools.

Grant Gate Exemptions

Approvals and gates can be temporarily suspended whenever an emergency deployment is required.

Approve Deployments

An approval is a mechanism used to control environments, regardless of quality (status) considerations. Approvals are attached to phases. A release that requires approval cannot proceed with a phase until approval is granted. Approvers are usually designated by user role. Any user with the designated role can respond to an approval request. If, for example, the QA phase requires approval by the release manager, the release will not be able to proceed until approval is granted by someone with the release manager role. Specific users can also be designated to approve.

Note

If a scheduled deployment requiring approval reaches its start time without having received approval, the phase will not proceed and will be considered rejected by the approver.

Execute deployments

A deployment is executed by running the tasks defined in the deployment's segments. The default deployment plan has two segments—Pre-Deployment Tasks, and Deployment Tasks. Segments are started manually, and, like tasks, can be run in parallel or serially. Start times are calculated from durations defined when the tasks were created.

Segments are displayed on the Scheduled Deployment page (*`Home > Scheduled Deployments > selected_deployment`*). The order in which segments are displayed is not the order in which they are executed.

Segments can be added or deleted at any time, and can have other segments designated as prerequisites. A segment with prerequisites cannot commence until the designated segments successfully complete all their tasks. Segments can also be restricted by user role; only users with the specified executor role can initiate the segment.

Users assigned to tasks can perform them, and tasks without an assigned user can be claimed by any user with the proper role.

Once you have scheduled the deployment, it will be added to the Scheduled Deployments page. There, you can edit, delete, or investigate the deployment.

Complete milestones

Update milestone status upon completion. Milestones extra-deployment items that can represent anything related to the release; their purpose is entirely up to you.

Execute the Production Deployment

After pre-production environments are successfully deployed, you can deploy to production.

Verify Segment Prerequisites

Ensure that every segment designated as a prerequisite for other segments has completed successfully. A segment, including those running in parallel with other segments, cannot commence until its prerequisite segments successfully complete. All segments can have prerequisites except the first one.

Segments can also be restricted by user role; only users with the specified executor role can initiate the segment.

Verify Overall Schedule

Ensure that all tasks have the expected duration; segment length is calculated using task durations. The order in which segments are displayed is not the order in which they are executed. If tasks are missing or require modification, make sure any new durations remain within the deployment's expected scope.

If approvals are required, make sure expected approvers are available and ready to act.

Segments are started manually, and, like tasks, can be run in parallel or serially.

Assign or Claim Tasks

Tasks can be assigned to a role or to a specific user. Unassigned tasks can be claimed by anyone with the defined role.

Configure Notifications

Email notifications can be sent to users whenever user-defined trigger events occur. Notifications can be attached to plan, segment, or task, and set to trigger in several ways. Notifications can be sent when a deployment finishes or an approval is required, for example. Notification recipients are defined with the security system's (see uDeploy Security) LDAP integration.

When setting up notifications, you select both the triggering events and the role, which is inherited from the security system. For example, it is common for an administrator or environment owner to be notified when a task or segment starts, ends, or experiences an error.

Monitor Deployments

The dashboard provides real-time feedback about your release from a centralized portal. It displays a release's segments and indicates the segment currently in progress, as well as those already completed. In addition to executing a deployment, you can add segments to the plan, and tasks to the segments.

Team- and Role-Based Security

uRelease's role- and team-based security system enables you to model releases that mimic your internal organization. Roles logically represent user functions and have actions defined to them, such as editing release plans or scheduling deployments.

Actions are defined per resource type. A resource type is a part of the product that can have user actions restricted by role. uRelease has several resource types, such as release, and life-cycle model. You define the actions a role can take for each resource type when you create the role. A role can have any or all actions granted to it.

Teams are collections of users grouped around a shared activity. Teams have access to particular resources (taken from the various resource types).

When a roles is assigned to a team, its actions are granted to the team. If a role grants the edit action to the project resource type, then the team can edit any project to which it has access.

Finally, before a user can perform any meaningful activity, he or she is assigned to a role on a team. The role defines the actions available to the user, and team membership determines the actual resources on which the actions can be applied.

uRelease Security

uRelease provides a flexible, role-based security model that maps to your organizational structure. Different product areas, such as components, can be secured by roles. Each area has a set of permissions available to it. To configure security for an area, you create roles using the available permissions—execute, read, write, and so forth.

So, how are permissions applied to users? First, global default permissions can be granted. Default permissions are granted by product area and apply to all users. If default permissions are granted for, say, the agent area, a user will have those permissions even if she is also part of a group or role that does not.

Another way users can be granted permissions is by being a member of a group. Groups can have default permissions that apply to all group members. If a user is assigned to a group with default permissions for the agent area, as above, she will have those permissions even if she is also assigned a role that does not have them.

Finally, users can be assigned to roles. Role members inherit a role's permissions. Except for UI and system security, users are assigned to roles on an item by item basis. For example, a user can be assigned a role that enables them to see only one application or only one component. Both groups and individual users can be assigned to roles.

Roles and permissions, including default permissions, are configured on an area by area basis; granting the execute permission to one role does not grant it to another. The default admin role has all permissions, but you can create another user with all permissions by creating a role for each area with all permissions granted, then assigning the user to each role. Typically, new roles are added to product areas during setup and occasionally thereafter.

While any number of roles can be created for an area, areas themselves cannot be created, modified (the available pool of permissions cannot be changed), or deleted.

Generally, you perform the following steps in order when setting-up security:

1. **Create Roles** Create roles and define permissions for the various product areas. For most evaluations, the default roles should be adequate.

   Use the UI security area to quickly assign access permissions to the different areas of uRelease.

   Use the system security area to assign usage permissions, including the ability to define security for other users.

2. **Authorization Realms**. Authorization realms are used by authentication realms to associate users with groups and to determine user access. uRelease includes both an internal database for storing security information as well as integration with the Lightweight Directory Access Protocol (*LDAP*). LDAP is a widely-used protocol for accessing distributed directory information over IP networks. If you are implementing a production version of uRelease, the LDAP integration is recommended. If you are evaluating uRelease, it is not necessary to set up the LDAP integration—full security is configured and enforced by the server.

3. **Create Groups and Define Default Permissions.** Determine default permissions by product area. Global default permissions can be granted.

4. **Create Authentication Realm**. The authentication realm is used to determine a user's identity within an authorization realm. If more than on realm has been configured, user authentication is determined following the hierarchy of realms defined on the Authentication pane. When a user attempts to log in, all realms are polled for matching credentials.

5. **Add Users.** Add users to an authentication realm, then assign them to groups and roles. If your are using LDAP, you can import users and map them to the security system.

Installing and Integrating

Installation

uRelease is configured as a Tomcat web application.

Server Installation Steps

1. Download and install the Apache Tomcat web server. Versions 6 and 7 are supported.

   ### Note

   Installing the Tomcat server on the same machine where uDeploy is located will improve performance for uDeploy integration. If you place both servers on the same machine, ensure that they do not use the same port.

2. Download uRelease from the UrbanCode support site: http://support.urbancode.com/. The installation package consists of a single `.war` file.

   ### Note

   The `.war` file's name will include the a version number. Because the file name will become part of the URL created by Tomcat, a good strategy is to rename the file to something meaningful to you and something that will not frequently change.

3. Deploy uRelease into Tomcat by copying the installation `.war` file to Tomcat's `webapps` directory. Before copying the file, first shutdown the Tomcat server. Tomcat will deploy the application the next time the server is started.

   ### Note

   If the Tomcat `autoDeploy` attribute is set to 'true,' the `.war` can be copied to Tomcat while it is running.

   Or, if you prefer, you can also use the Tomcat Application Manager to deploy the `.war` file.

Once deployed, you can access uRelease at the following URL: `http://host:port/urelease_name`

where `host:port` resolves to your Tomcat instance, and `urelease_name` is name of the uRelease `.war` file.

The initial credentials are:

- user ID: *admin*
- password: *admin*

Database Configuration

Out of the box, uRelease uses the Derby database--no additional user actions are required. uRelease can also be used with Oracle, MySQL or MS SQL Server.

Database configuration is done by modifying parameters in the *conf* file, which is located at *user_home_directory*/.urbancode/urelease/conf. On Unix, the *user_home_directory* is */home*; on Windows it is determined by the *user profile*, which typically is \Users*current_user*.

Because uRelease is deployed as a Tomcat web application, the *conf* file will not be created until uRelease is deployed, which means that the configuration changes cannot be made until after initial deployment.

To change the database, first deploy uRelease as described above, then shutdown down the Tomcat server. Next, modify the *conf* file following the instructions below for your database type. Once configured, restart Tomcat.

Configuring MySQL

Before installing the uRelease server, create a MySQL database. Ideally the database will be installed on a machine other than the one where the server is located.

When you install uRelease, you will need the connection information, and a user account with table creation privileges.

1. Create a MySQL database named *uRelease.*
2. Create a user named *uRelease* and grant it all privileges on the database. The user password will be used in the *conf* file.
3. Obtain the MySQL JDBC driver .jar file. The driver is unique to the edition you are using.

   Copy the JDBC .jar file to *tomcat_directory*\lib.
4. Modify the following parameters in the *conf* file:

```
database.type=mysql
hibernate.connection.driver_class=com.mysql.jdbc.Driver
hibernate.connection.url=jdbc\:mysql\://localhost\:3306/urelease
hibernate.connection.username=urelease
hibernate.connection.password=urelease
```

   ## Note

   Initially, the password contains an encrypted string. Overwrite the string with the new password. The first time the password is entered it will be plain text, but will be encrypted the next time uRelease starts.
5. When you are fished, restart the Tomcat server.

Configuring Oracle

Before installing the uRelease server, create an Oracle database. Ideally the database will be installed on a machine other than the one where the server is located.

When you install uRelease, you will need the connection information, and a user account with at least these minimum privileges:

RESOURCE, CONNECT, CREATE SESSION, and *CREATE TABLE*.

uRelease supports the following editions:

- Oracle Database Enterprise
- Oracle Database Standard
- Oracle Database Standard One
- Oracle Database Express

Version 10g or later is supported for each edition.

To install the database

1. Obtain the Oracle JDBC driver. The JDBC jar file is included among the Oracle installation files. The driver is unique to the edition you are using.

   Copy the JDBC jar file to *uRelease_installer_directory*\lib\ext.

2. Modify the following parameters in the *conf* file:

   ```
   database.type=oracle
   hibernate.connection.driver_class=oracle.jdbc.driver.OracleDriver
   hibernate.connection.username=urelease
   hibernate.connection.password=urelease
   hibernate.connection.url=jdbc\:mysql\://localhost\:3306/urelease
   ```

   A typical *hibernate.connection.url* might be:

   jdbc:oracle:thin:@localhost:1521:XE

   > **Note**
   >
   > Initially, the password contains an encrypted string. Overwrite the string with the new password. The first time the password is entered it will be plain text, but will be encrypted the next time uRelease starts.

3. When you are fished, restart the Tomcat server.

   > **Note**
   >
   > The database schema to use should be set as the user's default schema.

MS SQL Server

Before installing the uRelease server, create an SQL Server database. Ideally the database will be installed on a machine other than the one where the server is located.

When you install uRelease, you will need the connection information, and a user account with table creation privileges.

To install the database

1. Install the database using the following parameters:

   ```
   CREATE DATABASE uRelease;

   USE uRelease;
   CREATE LOGIN uRelease WITH PASSWORD = 'password';
   ```

```
CREATE USER uRelease FOR LOGIN uRelease WITH DEFAULT_SCHEMA = uRelease;
CREATE SCHEMA uRelease AUTHORIZATION uRelease;
GRANT ALL TO uRelease;
```

2. Obtain the SQL Server JDBC driver from the Microsoft site. The JDBC jar file is not included among the installation files.

   Copy the JDBC jar file to *uRelease_installer_directory*\lib\ext.

3. Modify the following parameters in the *conf* file:

```
database.type=sqlserver
hibernate.connection.driver_class=
 com.microsoft.sqlserver.jdbc.SQLServerDriver
hibernate.connection.username=urelease
hibernate.connection.password=urelease
hibernate.connection.url=
 jdbc:sqlserver://[DB_URL]:[DB_PORT];databaseName=[DB_NAME]
```

   A typical *hibernate.connection.url* might be:

   jdbc:sqlserver://localhost:1433;databaseName=uRelease

   > **Note**
   >
   > Initially, the password contains an encrypted string. Overwrite the string with the new password. The first time the password is entered it will be plain text, but will be encrypted the next time uRelease starts.

4. When you are fished, restart the Tomcat server.

Integrations

Integration refers to uRelease using and managing the objects and artifacts produced by external tools, such as uDeploy. Once successfully integrated with uDeploy, for example, its components and applications are automatically available to uRelease.

uDeploy Integration

To configure a uDeploy integration, add an entry on the Integration Providers page (*Home > Integrations)*, and provide the uDeploy URL and authentication token (preferably for the admin user).

Once configured, uRelease will periodically (once per minute) check uDeploy. Any applications or components that are found will be imported and displayed on the Applications page. If expected applications and components fail to appear, check the Tomcat log files for error messages.

> **Note**
>
> Authentication tokens are created with uDeploy's Token page (*Settings > Tokens*. Copy the generated token string and paste it into the Admin Auth Token field.

Jira Integration

Jira projects are mapped to uRelease initiatives. All issues in a mapped project are represented by change items in uRelease.

To configure a Jira integration, add an entry on the Integration Providers page (*Home > Integrations)*, and provide the Jira URL, user name, and password.

Once integrated, all Jira projects are available. To map a project, select it from the drop-down list box associated with the target initiative. Any issues in the mapped project will be imported as changes. Jira bug-type issues become bug-type changes in uRelease. All other Jira issue types become other-type changes.

Note

Jira projects cannot be imported unless they are mapped to intiatives.

Index