Version 28.0: Summer '13

Force.com Migration Tool Guide

Last updated: June 29, 2013

Table of Contents

Chapter 1

Force.com Migration Tool Overview

The Force.com Migration Tool is a Java/Ant-based command-line utility for moving metadata between a local directory and a Salesforce organization. The Force.com Migration Tool is especially useful in the following scenarios:

- Development projects where you need to populate a test environment with large amounts of setup changes — Making these changes using a Web interface could take a long time.
- Multistage release processes — A typical development process requires iterative building, testing, and staging before releasing to a production environment. Scripted retrieval and deployment of components can make this process much more efficient.
- Repetitive deployment using the same parameters — You can retrieve all the metadata in your organization, make changes, and deploy a subset of components. If you need to repeat this process, it's as simple as calling the same deployment target again.
- When migrating from stage to production is done by IT — Anyone that prefers deploying in a scripting environment will find the Force.com Migration Tool a familiar process.

Understanding Metadata API

Metadata API contains a set of objects that manage setup and customization information (metadata) for your organizations, and the SOAP calls that manipulate those objects. With Metadata API you can:

- Work with setup configuration as XML metadata files
- Migrate configuration changes between organizations
- Create your own tools for managing organization and application metadata

Though you could write your own client applications for using Metadata API SOAP calls, Salesforce provides the Force.com Migration Tool to retrieve and deploy Apex and metadata.

Understanding Package and Directory Structure

Metadata API functions in a package-centric manner. Components may be in one or more packages, or in no package at all. Packages may be local (created in your Salesforce organization) or installed from Force.com AppExchange. Whenever the Force.com Migration Tool retrieves a set of components, that set will be limited to what's in a single package or what's in no package at all. There are three kinds of packages:

- Unpackaged—Components that live natively in your organization, such as standard objects, go in the *unpackaged* package.
- Unmanaged package—Unmanaged packages are typically used to distribute open-source projects or application templates to provide developers with the basic building blocks for an application. Once the components are installed from an unmanaged package, the components can be edited in the organization they are installed in. The developer who created and uploaded the unmanaged package has no control over the installed components, and can't change or upgrade them. Unmanaged packages should not be used to migrate components from a sandbox to production organization. Instead, use Change Sets.
- Managed package—A collection of application components that is posted as a unit on the AppExchange and associated with a namespace and possibly a License Management Organization. To support upgrades, a package must be managed. An organization can create a single managed package that can be downloaded and installed by many different organizations. Managed packages differ from unmanaged packages by having some locked components, allowing the managed package to be upgraded later. Unmanaged packages do not include locked components and cannot be upgraded. In addition, managed

packages obfuscate certain components (like Apex) on subscribing organizations to protect the intellectual property of the developer.

Chapter 2

Installing the Force.com Migration Tool

Before you install the Force.com Migration Tool you will need Java and Ant installed on your local machine. Then you can download the Force.com Migration Tool from a Salesforce organization.

1. Install Java and Ant, as described in Prerequisites for Using the Force.com Migration Tool.
2. Log into a Salesforce organization and download the Force.com Migration Tool, as described in Installing the Force.com Migration Tool.

Prerequisites for Using the Force.com Migration Tool

Before you can use the Force.com Migration Tool you must have Java and Ant installed and configured correctly. You may not need to install Java and Ant if you already have them on your computer, so first verify the installation from a command prompt.

Java

To see if you have Java installed:

1. Open a command prompt.
2. At the prompt, type `java -version` and press Enter.

The output should look something like the following:

```
java version "1.6.0_39"
Java(TM) SE Runtime Environment (build 1.6.0_39-b04)
Java HotSpot(TM) 64-Bit Server VM (build 20.14-b01, mixed mode)
```

The Force.com Migration Tool works with Java version 1.6.x or later. If you have an earlier version, you'll need to install Java 1.6.x or later.

To install Java, go to http://www.oracle.com/technetwork/java/javase/downloads/index.html and get the latest version of the Java JDK. When you're finished with the installation, verify by typing `java -version` at a command prompt.

Ant

To see if you have Ant installed:

1. Open a command prompt.
2. At the prompt, type `ant -version` and press Enter.

The output should look something like the following:

```
Apache Ant version 1.7.0 compiled on December 13 2006
```

If the Ant version is 1.5.x or earlier, you will need to download the latest version of Ant.

Note: Even if you have Ant installed, you may still need to put the `bin` directory on your path. On a Windows operation system, you may also need to set the `ANT_HOME` and `JAVA_HOME` environment variables as follows.

To install and configure Ant:

1. Download Apache Ant version 1.6 or newer to a directory of your choice: `http://ant.apache.org/bindownload.cgi`. This directory will be known as ANT_HOME. Once the files are on your computer, there is no further installation required.
2. Add the `bin` directory to your path. (Only the `bin` and `lib` directories are required to run Ant.)
3. If you are using a Windows operation system, create an `ANT_HOME` environment variable and set the value to where you have installed Ant. Also create a `JAVA_HOME` environment variable and set the value to the location of your JDK.

For additional information, see `http://ant.apache.org/manual/install.html`.

Installing the Force.com Migration Tool

To download and install the Force.com Migration Tool:

1. Log into a Salesforce organization on your deployment machine.
2. From Setup, click **Develop** > **Tools**, and then click **Force.com Migration Tool**.
3. Save the `.zip` file locally and extract the contents to the directory of your choice.
4. Copy `ant-salesforce.jar` and paste into your Ant installation's `lib` directory. The `lib` directory is located in the root folder of your Ant installation.

Note: If you do not have Ant installed, see Prerequisites for Using the Force.com Migration Tool.

When you extract the Force.com Migration Tool zip files, the following folders and files are written to the location you specified:

- A `Readme.html` file that explains how to use the tools
- A Jar file containing the ant task: `ant-salesforce.jar`
- A sample folder containing:
  - A `codepkg\classes` folder that contains `SampleDeployClass.cls` and `SampleFailingTestClass.cls`
  - A `codepkg\triggers` folder that contains `SampleAccountTrigger.trigger`
  - A `mypkg\objects` folder that contains the custom objects used in the examples
  - A `removecodepkg` folder that contains XML files for removing the examples from your organization
  - A sample `build.properties` file that you must edit, specifying your credentials, in order to run the sample ant tasks in `build.xml`
  - A sample `build.xml` file, that exercises the `deploy` and `retrieve` API calls

Chapter 3

Using the Force.com Migration Tool

The Force.com Migration Tool is a Java/Ant-based command-line utility for moving metadata between a local directory and a Salesforce organization. You can use the Force.com Migration Tool to retrieve components, create scripted deployment, and repeat deployment patterns.

The general procedure you will follow when using the Force.com Migration Tool to copy metadata from one Salesforce organization to another is:

1. Enter credentials and connection information for source Salesforce organization in `build.properties`
2. Create retrieve targets in `build.xml`
3. Construct a project manifest in `package.xml`
4. Run the Force.com Migration Tool to retrieve metadata files from Salesforce
5. Enter credentials and connection information for destination Salesforce organization in `build.properties`
6. Run the Force.com Migration Tool to deploy metadata files or deletions to Salesforce

Entering Salesforce Connection Information

In order to retrieve or deploy metadata components, you need to edit `build.properties` to point to a Salesforce organization:

1. Go to the location where you extracted the Force.com Migration Tool files and open the `sample` subdirectory.
2. Open `build.properties` in a text editor and substitute a valid Salesforce username and password. If you are using a security token, paste the 25-digit token value to the end of your password.

Parameter	Value
`sf.username`	The salesforce.com username for login. The username associated with this connection must have the "Modify All Data" permission. Typically, this is only enabled for System Administrator users. When connecting to a sandbox instance your sandbox name is appended to your username. For example, if your production username is `foo@salesforce.com`, and one of your sandboxes is called `bar`, then your sandbox username is `foo@salesforce.com.bar`.
`sf.password`	The password you use to log into the organization associated with this project. If you are using a security token, paste the 25-digit token value to the end of your password.
`sf.serverurl`	The salesforce server URL. Use `https://login.salesforce.com` to connect to a production or Developer Edition organization. To connect to a sandbox instance, change this to `https://test.salesforce.com`.

Constructing a Project Manifest

The `package.xml` file is a project manifest that lists all the components you want to retrieve or deploy in a single request. You can retrieve or deploy only a single package at a time.

The following elements may be defined in `package.xml`:

Name	Description
`<fullName>`	The name of the server-side package to deploy into. If the `<fullName>` field is omitted, components will not be assigned to a package when deployed, and will be in the `unpackaged` package. This field is not used for retrieve.
`<types>`	This element contains one or more `<members>` tags and one `<name>` tag, and is used to list the metadata components of a certain type to retrieve or deploy.
`<members>`	The full name of a component. There is one `<members>` element defined for each component in the directory. You can replace the value in this member with the wildcard character `*` (asterisk) instead of listing each member separately. This is a child element of `<types>`.
`<name>`	Contains the type of the component, for example `CustomObject` or `Profile`. There is one name defined for each component type in the directory. This is a child element of `<types>`.
`<version>`	The Metadata API version number of the files being retrieved or deployed. When deploying, all the files must conform to the same version of the Metadata API.

Component Types

The following table lists the component types that can be defined by the `<name>` element in `package.xml`:

Component	XML <name> Metadata Type	Folder	Uses *	Notes
Account Criteria Based Sharing Rule	`criteriaBasedRules`	`accountSharingRules`	yes	AccountCriteriaBasedSharingRule is represented as `criteriaBasedRules`, contained in the AccountSharingRules component.
Account Owner Sharing Rule	`ownerRules`	`accountSharingRules`	yes	AccountOwnerSharingRule is represented as `ownerRules`, contained in the AccountSharingRules component.
Account Territory Sharing Rule	`rules`	`accountTerritorySharingRules`	yes	AccountTerritorySharingRule is represented as `rules`, contained in the AccountTerritorySharingRules component.
Action Override	`ActionOverride`	`objects`	no	This type is retrieved or deployed as part of an object file. You must dot-qualify

Component	XML <name> Metadata Type	Folder	Uses *	Notes
				the object name before the component name. You can only access ActionOverride by accessing its encompassing CustomObject.
Activity Settings	ActivitiesSettings	`settings`	yes	Represents an organization's activity settings, and its user interface settings for the calendar.
Address Settings	`Settings`	`settings`	yes	Represents the configuration of country and state picklists. Country and state picklists and the AddressSettings metadata type are in beta release.
Analytic Snapshot	`AnalyticSnapshot`	`analyticsnapshots`	no	
Apex Class	`ApexClass`	`classes`	yes	
Approval Process	`ApprovalProcess`	`approvalProcesses`	yes	Supported in change sets. Not supported in managed or unmanaged packages. You can use the wildcard (*) symbol to retrieve all approval processes for all objects. You can't use it to retrieve a subset of approval processes; syntax such as `Lead.*` is not supported.
Article Type	`ArticleType`	`objects`	yes	
Apex Trigger	`ApexTrigger`	`triggers`	yes	
Assignment Rules	`AssignmentRules`	`assignmentRules`	yes	For accessing individual rules for a particular object, use `AssignmentRule` instead of `AssignmentRules`.
Authentication Provider	`AuthProvider`	`authproviders`	no	
Auto-Response Rules	`AutoResponseRules`	autoResponseRules	yes	For accessing individual rules for a particular object, use `AutoResponseRule` instead of `AutoResponseRules`.
Business Process	`BusinessProcess`	`objects`	no	This type is retrieved or deployed as part of an object file. You must dot-qualify the object name before the component name.
Call Center	`CallCenter`	`callCenters`	yes	

Component	XML <name> Metadata Type	Folder	Uses *	Notes
Campaign Criteria Based Sharing Rule	`criteriaBasedRules`	`campaignSharingRules`	yes	CampaignCriteriaBasedSharingRule is represented as `criteriaBasedRules`, contained in the CampaignSharingRules component.
Case Settings	`Settings`	`settings`	yes	
Campaign Owner Sharing Rule	`ownerRules`	`campaignSharingRules`	yes	CampaignOwnerSharingRule is represented as `ownerRules`, contained in the CampaignSharingRules component.
Company Settings	`Settings`	`settings`	yes	
Case Criteria Based Sharing Rule	`criteriaBasedRules`	`caseSharingRules`	yes	CaseCriteriaBasedSharingRule is represented as `criteriaBasedRules`, contained in the CaseSharingRules component.
Case Owner Sharing Rule	`ownerRules`	`caseSharingRules`	yes	CaseOwnerSharingRule is represented as `ownerRules`, contained in the CaseSharingRules component.
Chatter Answers Settings	`Settings`	`settings`	yes	
Community	`Community (Zone)`	`communities`	yes	
Contact Criteria Based Sharing Rule	`criteriaBasedRules`	`contactSharingRules`	yes	ContactCriteriaBasedSharingRule is represented as `criteriaBasedRules`, contained in the ContactSharingRules component.
Contact Owner Sharing Rule	`ownerRules`	`contactSharingRules`	yes	ContactOwnerSharingRule is represented as `ownerRules`, contained in the ContactSharingRules component.
Contract Settings	`Settings`	`settings`	yes	
Custom Object Criteria Based Sharing Rule	`criteriaBasedRules`	`customObjectSharingRules`	no	CustomObjectCriteriaBasedSharingRule is represented as `criteriaBasedRules`, contained in the CustomObjectSharingRules component.
Custom Object Owner Sharing Rule	`ownerRules`	`customObjectSharingRules`	no	CustomObjectOwnerSharingRule is represented as `ownerRules`, contained in the CustomObjectSharingRules component.

Component	XML <name> Metadata Type	Folder	Uses *	Notes
Custom Application	`CustomApplication`	`applications`	yes	
Custom Field	`CustomField`	`objects`	no	Custom fields are retrieved or deployed as part of a custom object file. You must dot-qualify the object name before the component name. Individual custom fields cannot be retrieved with the wildcard (*) symbol, but must be explicitly named in `package.xml`, unless their object is named in the `CustomObject` section.
Custom Label	`CustomLabels`	`labels`	yes	Custom labels that can be localized for use in different languages, countries, and currencies.
Custom Object or Standard Object	`CustomObject`	`objects`	yes	Standard objects cannot be retrieved with the wildcard (*) symbol, but must be explicitly named in `package.xml`, and only custom fields and standard picklist fields are included.
Custom Object Translation	`CustomObjectTranslation`	`objectTranslations`	yes	
Custom Page Web Link	`CustomPageWebLink`	`weblinks`	yes	Web links are defined in a home page component.
Custom Site	`CustomSite`	`sites`	yes	
Custom Tab	`CustomTab`	`tabs`	yes	
Dashboard	`Dashboard`	`dashboards`	no	
Data Categories	`DataCategoryGroup`	`datacategorygroups`	yes	Includes a category group and its categories.
Document	`Document`	`document`	no	
Email Template	`EmailTemplate`	`email`	no	
Entitlement Process	`EntitlementProcess`	`entitlementProcess`	yes	
Entitlement Settings	`Settings`	`settings`	yes	
Entitlement Template	`EntitlementTemplate`	`entitlementTemplates`	yes	

Component	XML <name> Metadata Type	Folder	Uses *	Notes
Escalation Rules	`EscalationRules`	escalationRules	yes	For accessing individual rules for a particular object, use `EscalationRule` instead of `EscalationRules`.
Field Set	`FieldSet`	`objects`	yes	
Flow	`Flow`	`flows`	yes	A `.flow` file is an XML representation of a flow definition.
Folder	`Folder`	`documents`, `email`, `reports`, or `dashboards`	no	A folder can contain documents, email templates, reports, or dashboards. You must specify the folder type (Document, EmailTemplate, Report, or Dashboard) to retrieve or deploy.
FolderShare	`FolderShare`	`reports` or `dashboards`	no	Represents the settings for enhanced analytics folder sharing. Users can control access to reports or dashboards by giving others Viewer, Editor or Manager access to the folder that contains the report or dashboard.
ForecastingSettings	`Settings`	`Forecasting.settings` in the `settings`	yes	ForecastingSettings values are stored in a single file named `Forecasting.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components, as there is only one settings file for each settings component.
Group	`Group`	`groups`	yes	
Home Page Component	`HomePageComponent`	`homePageComponents`	yes	
Home Page Layout	`HomePageLayout`	`homePageLayouts`	yes	
Idea Settings	`Settings`	`settings`	yes	
Installed Package	`InstalledPackage`	`installedPackages`	yes	Represents a package to be installed or uninstalled. Deploying a newer version of a currently installed package upgrades the package.
Knowledge Settings	`Settings`	`settings`	yes	
Letterhead	`Letterhead`	`letterhead`	no	

Component	XML <name> Metadata Type	Folder	Uses *	Notes
List View	`ListView`	`objects`	no	This type is retrieved or deployed as part of an object file. You must dot-qualify the object name before the component name.
Live Agent Settings	`LiveAgentSettings`	`settings`	yes	
Live Agent Agent Configurations	`LiveChatAgentConfig`	`liveChatAgentConfig`	yes	
Live Agent Chat Button	`LiveChatButton`	`liveChatButton`	yes	
Live Agent Deployment	`LiveChatDeployment`	`liveChatDeployment`	yes	
Live Agent Skills	`Skill`	`skill`	yes	
Lookup Filter	`NamedFilter`	`objects`	no	This type is retrieved or deployed as part of an object file. You must dot-qualify the object name before the component name.
Lead Criteria Based Sharing Rule	`criteriaBasedRules`	`leadSharingRules`	yes	LeadCriteriaBasedSharingRule is represented as `criteriaBasedRules`, contained in the LeadSharingRules component.
Lead Owner Sharing Rule	`ownerRules`	`leadSharingRules`	yes	LeadOwnerSharingRule is represented as `ownerRules`, contained in the LeadSharingRules component.
Milestone Type	`MilestoneType`	`milestoneType`	yes	
Mobile Settings	`Settings`	`settings`	yes	
Network	`Network`	`networks`	yes	
Opportunity Criteria Based Sharing Rule	`criteriaBasedRules`	`opportunitySharingRules`	yes	OpportunityCriteriaBasedSharingRule is represented as `criteriaBasedRules`, contained in the OpportunitySharingRules component.
Opportunity Owner Sharing Rule	`ownerRules`	`opportunitySharingRules`	yes	OpportunityOwnerSharingRule is represented as `ownerRules`, contained in the OpportunitySharingRules component.
Opportunity Settings	`OpportunitySettings`	`settings`	no	

Component	XML <name> Metadata Type	Folder	Uses *	Notes
Page Layout	`Layout`	`layouts`	yes	
Permission Set	`PermissionSet`	`permissionsets`	yes	
Portal	`Portal`	`portals`	yes	
Product Settings	`ProductSettings`	`settings`	no	
Profile	`Profile`	`profiles`	yes	
Queue	`Queue`	`queues`	yes	
QuickAction	`QuickAction`	`quickAction`	yes	Represents a specified create or update action for an object that then becomes available in the Chatter publisher. For example, you can create an action that, on the detail page of an account, allows a user to create a contact related to that account from the Chatter feed on that page. QuickAction can be created on objects that allow custom fields.
Quote Settings	`QuoteSettings`	`settings`	no	
Record Type	`RecordType`	`objects`	yes	This type is retrieved or deployed as part of an object file. You must dot-qualify the object name before the component name.
Remote Site Setting	`RemoteSiteSetting`	`remoteSiteSettings`	yes	
Report	`Report`	`reports`	no	
Report Type	`ReportType`	`reportTypes`	yes	Custom report types allow you to build a framework from which users can create and customize reports.
Role	`Role`	`roles`	yes	
SAML Single Sign-On	`SamlSsoConfig`	`samlssoconfigs`	yes	
Scontrol	`Scontrol`	`scontrols`	yes	Deprecated. Use Visualforce pages.
Security Settings	`Settings`	`settings`	yes	
Sharing Reason	`SharingReason`	`objects`	no	This type is retrieved or deployed as part of an object file. You must dot-qualify the object name before the component name.
Sharing Recalculation	`SharingRecalculation`	`objects`	no	This type is retrieved or deployed as part of an object file. You must dot-qualify

Component	XML <name> Metadata Type	Folder	Uses *	Notes
				the object name before the component name.
Static Resource	StaticResource	staticResources	yes	
Territory	Territory	territories	yes	
Translation Workbench	Translations	translations	yes	
Validation Rule	ValidationRule	objects	no	This type is retrieved or deployed as part of an object file. You must dot-qualify the object name before the component name.
Visualforce Component	ApexComponent	components	yes	
Visualforce Page	ApexPage	pages	yes	
Web Link	Weblink	objects	no	This type is retrieved or deployed as part of an object file. You must dot-qualify the object name before the component name.
Workflow	Workflow	workflows	yes	A .workflow file is a container for the individual workflow components associated with an object.

Specifying Standard Objects

To retrieve standard objects and/or custom fields on standard objects, you must name the component in package.xml. The following package.xml file will retrieve a single field EngineeringReqNumber__c, on the Case object, as well as the entire Account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Case.EngineeringReqNumber__c</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>Account</members>
        <name>CustomObject</name>
    </types>
    <version>28.0</version>
</Package>
```

Note: Custom objects and standard objects should be specified in the same <types> section, the one containing <name>CustomObject</name>.

Specifying Named Components

To retrieve a component, specify the type of component in the `<name>` element and declare each component to be retrieved or deployed in the `<members>` element. The following is a sample `package.xml` project manifest that names two custom objects to be retrieved or deployed:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyCustomObject__c</members>
        <members>MyHelloWorldObject__c</members>
        <name>CustomObject</name>
    </types>
    <version>28.0</version>
</Package>
```

Some metadata components are sub-components of another component. This means you must dot-qualify the sub-component with the parent component name.

The following metadata components are defined as part of an object:

- `CustomField`
- `Picklist`
- `RecordType`
- `Weblink`
- `ValidationRule`

For example, the following code retrieves a validation rule called `ValidationRuleName` on the Opportunity object:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Opportunity.ValidationRuleName</members>
        <name>ValidationRule</name>
    </types>
    <version>28.0</version>
</Package>
```

Specifying all Components of a Type

To retrieve all components of a particular type, use the wildcard symbol (*). For example, to retrieve all custom objects:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CustomObject</name>
    </types>
    <version>28.0</version>
</Package>
```

The wildcard symbol does not apply to all metadata types. For example, using the wildcard with the `CustomObject` type name will not retrieve standard objects. To retrieve a standard object, you must explicitly name the object in `package.xml`. Likewise, if you want to retrieve custom fields defined on standard objects, you must name the object and field.

Specifying Standard Objects

To retrieve standard objects and/or custom fields on standard objects, you must name the component in `package.xml`. The following `package.xml` file will retrieve a single field `EngineeringReqNumber__c`, on the Case object, as well as the entire Account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Case.EngineeringReqNumber__c</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>Account</members>
        <name>CustomObject</name>
    </types>
    <version>28.0</version>
</Package>
```

Note: Custom objects and standard objects should be specified in the same `<types>` section, the one containing `<name>CustomObject</name>`.

Getting Information About Metadata Types

You sometimes need to experiment with the composition of your `package.xml` manifest file before you settle on the final version that retrieves or deploys the metadata that you want. There are a couple of helper targets, `<sf:describeMetadata>` and `<sf:listMetadata>`, that are useful for gathering the relevant information during this experimentation period. The `build.xml` file specifies a series of commands to be executed by Ant. Within the `build.xml` file are named targets that process a series of commands when you run Ant with a target name.

Describing Metadata Types

The `describeMetadata` target returns a list of metadata types that are enabled for your organization. This target is useful when you want to identify the syntax needed for a metadata type in a `<name>` element in `package.xml`; for example, `CustomObject` for custom objects or `Layout` for page layouts. The following parameters may be set for each `<sf:describeMetadata>` target:

Field	Description
`username`	Required. The Salesforce username for login. The username associated with this connection must have the “Modify All Data” permission. Typically, this is only enabled for System Administrator users.
`password`	Required. The password you use to log into the organization associated with this project. If you are using a security token, paste the 25-digit token value to the end of your password.
`serverurl`	Optional. The Salesforce server URL (if blank, defaults to `login.salesforce.com`). To connect to a sandbox instance, change this to `test.salesforce.com`.
`apiVersion`	Optional. The API version to use for the metadata. The default is 28.0.

Field	Description
resultFilePath	Optional. The path of the output file where results are stored. The default output is the console. Directing the output to a file makes it easier to extract the relevant information for your package.xml manifest file.
trace	Optional. Defaults to false. Prints the SOAP requests and responses to the console. Note that this will show the user's password in plain text during login.

To get the list of metadata types enabled for your organization, specify a target in the build.xml file using <sf:describeMetadata>.

```
<target name="describeMetadata">
  <sf:describeRetrieve
    username="${sf.username}"
    password="${sf.password}"
    serverurl="${sf.serverurl}"
    resultFilePath="describeMetadata/describe.log"/>
</target>
```

Listing Components for a Metadata Type

The listMetadata target retrieves property information about metadata components in your organization. This target is useful when you want to identify individual components in package.xml for a retrieval or if you want a high-level view of particular metadata types in your organization. For example, you could use this target to return a list of names of all the CustomObject or Layout components in your organization, and use this information to make a subsequent retrieval to return a subset of these components. The following parameters may be set for each <sf:listMetadata> target:

Field	Description
username	Required. The Salesforce username for login. The username associated with this connection must have the "Modify All Data" permission. Typically, this is only enabled for System Administrator users.
password	Required. The password you use to log into the organization associated with this project. If you are using a security token, paste the 25-digit token value to the end of your password.
serverurl	Optional. The Salesforce server URL (if blank, defaults to login.salesforce.com). To connect to a sandbox instance, change this to test.salesforce.com.
metadataType	Required. The name of the metadata type for which you are retrieving property information; for example, CustomObject for custom objects, or Report for custom reports. For a full list of allowed values, see Component Types on page 6.
folder	The folder associated with the component. This field is required for components that use folders, such as Dashboard, Document, EmailTemplate, or Report.
apiVersion	Optional. The API version to use for the metadata. The default is 28.0.
resultFilePath	Optional. The path of the output file where results are stored. The default output is the console. Directing the output to a file makes it easier to extract the relevant information for your package.xml manifest file.

Field	Description
trace	Optional. Defaults to false. Prints the SOAP requests and responses to the console. Note that this will show the user's password in plain text during login.

To get property information for components of one metadata type, such as CustomObject, specify a target in the build.xml file using <sf:listMetadata>.

```
<target name="listMetadata">
  <sf:listMetadata
    username="${sf.username}"
    password="${sf.password}"
    serverurl="${sf.serverurl}"
    metadataType="CustomObject"
    resultFilePath="listMetadata/list.log"/>
</target>
```

The following example uses a component that resides in a folder.

```
<target name="listMetadata">
  <sf:listMetadata
    username="${sf.username}"
    password="${sf.password}"
    serverurl="${sf.serverurl}"
    metadataType="Report"
    folder="Marketing_Reports"
    resultFilePath="listMetadata/list.log"/>
</target>
```

Creating Retrieve Targets

The build.xml file specifies a series of commands to be executed by Ant. Within the build.xml file are named targets that process a series of commands when you run Ant with a target name. The following parameters may be set for each <sf:retrieve> target:

Field	Description
username	Required. The Salesforce username for login. The username associated with this connection must have the "Modify All Data" permission. Typically, this is only enabled for System Administrator users.
password	Required. The password you use to log into the organization associated with this project. If you are using a security token, paste the 25-digit token value to the end of your password.
serverurl	Optional. The Salesforce server URL (if blank, defaults to login.salesforce.com). To connect to a sandbox instance, change this to test.salesforce.com.
retrieveTarget	Required. The root of the directory structure into which the metadata files are retrieved.

Field	Description
packageNames	Required if unpackaged is not specified. A comma-separated list of the names of the packages to retrieve. You must specify either packageNames or unpackaged, but not both.
apiVersion	Optional. The Metadata API version to use for the retrieved metadata files. The default is 28.0.
pollWaitMillis	Optional. Defaults to 10000. The number of milliseconds to wait between attempts when polling for results of the retrieve request. The client continues to poll the server up to the limit defined by maxPoll.
maxPoll	Optional. Defaults to 20. The number of times to poll the server for the results of the retrieve request. The wait time between successive poll attempts is defined by pollWaitMillis.
singlePackage	Optional. Defaults to true. This must be set to false if you are retrieving multiple packages. If set to false, the retrieved zip file includes an extra top-level directory containing a subdirectory for each package.
trace	Optional. Defaults to false. Prints the SOAP requests and responses to the console. Note that this will show the user's password in plain text during login.
unpackaged	Required if packageNames is not specified. The path and name of a file manifest that specifies the components to retrieve. You must specify either unpackaged or packageNames, but not both.
unzip	Optional. Defaults to true. If set to true, the retrieved components are unzipped. If set to false, the retrieved components are saved as a zip file in the retrieveTarget directory.

Retrieving Components in Bulk

This target is the optimal way to download a large number of components of a single metadata type, such as custom reports, into a set of local files. The following parameters may be set for each <sf:bulkRetrieve> target:

Field	Description
username	Required. The Salesforce username for login. The username associated with this connection must have the “Modify All Data” permission. Typically, this is only enabled for System Administrator users.
password	Required. The password you use to log into the organization associated with this project. If you are using a security token, paste the 25-digit token value to the end of your password.
serverurl	Optional. The Salesforce server URL (if blank, defaults to login.salesforce.com). To connect to a sandbox instance, change this to test.salesforce.com.

Field	Description
`retrieveTarget`	Required. The root of the directory structure into which the metadata files are retrieved.
`metadataType`	Required. The name of the metadata type to be retrieved; for example, `CustomObject` for custom objects, or `Report` for custom reports. For a full list of allowed values, see Component Types on page 6.
`containingFolder`	Optional. If the metadata is contained in a folder, this parameter should be the name of the folder from which the contents are retrieved.
`batchSize`	Optional, defaults to 10. The number of items to retrieve while doing multi-part retrieve.
`apiVersion`	Optional. The Metadata API version to use for the retrieved metadata files. The default is 28.0.
`maxPoll`	Optional. Defaults to `20`. The number of times to poll the server for the results of the retrieve request. The clients waits for two seconds after the first poll attempt. The wait time is doubled for each successive poll attempt up to maximum of 30 seconds between poll attempts.
`unzip`	Optional. Defaults to `true`. If set to `true`, the retrieved components are unzipped. If set to `false`, the retrieved components are saved as a zip file in the `retrieveTarget` directory.

To retrieve custom report components in bulk, specify a target in the `build.xml` file using `<sf:bulkRetrieve>`.

```
<target name="bulkRetrieve">
  <sf:bulkRetrieve
    username="${sf.username}"
    password="${sf.password}"
    serverurl="${sf.serverurl}"
    metadataType="Report"
    retrieveTarget="retrieveUnpackaged"/>
</target>
```

Retrieving Unpackaged Components

The *unpackaged* package contains all of the standard objects, custom objects, Apex classes and other metadata components that exist natively in your organization, and not within a package. To retrieve unpackaged components, use a `build.xml` target that contains the unpackaged attribute that points to a `package.xml` file. For example:

```
<target name="retrieveUnpackaged">
  <mkdir dir="projectFolder"/>
  <sf:retrieve
      username="${sf.username}"
      password="${sf.password}"
      serverurl="${sf.serverurl}"
      retrieveTarget="projectFolder"
      unpackaged="unpackaged/package.xml"/>
</target>
```

The `salesforce-ant.jar` file contains Ant *tasks* for accessing the Metadata API. In the code above, `sf:retrieve` is an Ant task. The full list of metadata Ant tasks are described in the Metadata API Developer's Guide.

Retrieving Managed or Unmanaged Packages

Packages are useful for distributing related bundles of metadata across multiple instances or organizations, via Force.com AppExchange. However, you can use the Force.com Migration Tool to freely retrieve and deploy packaged metadata without using AppExchange. You retrieve both managed and unmanaged packages in the same way.

To retrieve a package, specify a `packageNames` parameter in the `build.xml` file. For example:

```
<target name="retrieveNamedPackage">
  <sf:retrieve
      username="${sf.username}"
      password="${sf.password}"
      serverurl="${sf.serverurl}"
      retrieveTarget="projectFolder"
      packageNames="mySourcePackage"/>
</target>
```

Retrieving Components in Bulk

This target is the optimal way to download a large number of components of a single metadata type, such as custom reports, into a set of local files. The following parameters may be set for each `<sf:bulkRetrieve>` target:

Field	Description
`username`	Required. The Salesforce username for login. The username associated with this connection must have the “Modify All Data” permission. Typically, this is only enabled for System Administrator users.
`password`	Required. The password you use to log into the organization associated with this project. If you are using a security token, paste the 25-digit token value to the end of your password.
`serverurl`	Optional. The Salesforce server URL (if blank, defaults to `login.salesforce.com`). To connect to a sandbox instance, change this to `test.salesforce.com`.
`retrieveTarget`	Required. The root of the directory structure into which the metadata files are retrieved.
`metadataType`	Required. The name of the metadata type to be retrieved; for example, `CustomObject` for custom objects, or `Report` for custom reports. For a full list of allowed values, see Component Types on page 6.
`containingFolder`	Optional. If the metadata is contained in a folder, this parameter should be the name of the folder from which the contents are retrieved.
`batchSize`	Optional, defaults to 10. The number of items to retrieve while doing multi-part retrieve.
`apiVersion`	Optional. The Metadata API version to use for the retrieved metadata files. The default is 28.0.

Field	Description
maxPoll	Optional. Defaults to 20. The number of times to poll the server for the results of the retrieve request. The clients waits for two seconds after the first poll attempt. The wait time is doubled for each successive poll attempt up to maximum of 30 seconds between poll attempts.
unzip	Optional. Defaults to true. If set to true, the retrieved components are unzipped. If set to false, the retrieved components are saved as a zip file in the retrieveTarget directory.

To retrieve custom report components in bulk, specify a target in the build.xml file using <sf:bulkRetrieve>.

```
<target name="bulkRetrieve">
  <sf:bulkRetrieve
    username="${sf.username}"
    password="${sf.password}"
    serverurl="${sf.serverurl}"
    metadataType="Report"
    retrieveTarget="retrieveUnpackaged"/>
</target>
```

Retrieving Metadata from a Salesforce Organization

To retrieve Force.com components:

1. Open a command prompt.
2. Run Ant by specifying a target name in build.xml. If this is the first time you are running Ant, use ant retrieveUnpackaged to retrieve unpackaged components specified in package.xml.

Note:

- The sample build.xml contains a number of useful targets for various retrieve() and deploy() options that you can modify or use as is. To see a list of all of your named targets in build.xml, enter ant -p at the command line.
- Metadata API can deploy and retrieve up to 5,000 files at one time. If you are working with a large number of components, you should use the listMetadata target to identify the subset of files that you want to retrieve, or you should retrieve batches of components using the bulkRetrieve target.

Editing Metadata

You can use any UTF-8 text editor to make changes to the files you retrieve.

Warning: Text editors that do not natively support UTF-8 may insert a byte order mark (BOM) at the top of the file, which can cause problems in the XML metadata.

Deleting Files from an Organization

The `package.xml` file is a project manifest that lists all the components you want to retrieve or deploy. While you can use `package.xml` to add components, it's not sufficient to delete them. To delete files, you must also create a delete manifest called `destructiveChanges.xml`. The format of the delete manifest is the same as `package.xml`, except that wildcards are not supported.

Note: If you try to delete components that do not exist in the organization, the rest of the deletions will be attempted.

The following is a sample `destructiveChanges.xml` file that names a single custom object to be deleted:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyCustomObject__c</members>
        <name>CustomObject</name>
    </types>
</Package>
```

In order to deploy the destructive changes, you must also have a `package.xml` file that lists no components to deploy, includes the API version, and is in the same directory as `destructiveChanges.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <version>28.0</version>
</Package>
```

Deploying Changes to a Salesforce Organization

The `build.xml` file specifies targets to retrieve and deploy. The following parameters may be set for each deploy target:

Field	Description
`username`	Required. The Salesforce username for login. The username associated with this connection must have the "Modify All Data" permission. Typically, this is only enabled for System Administrator users.
`password`	Required. The password you use to log into the organization associated with this project. If you are using a security token, paste the 25-digit token value to the end of your password.
`serverurl`	Optional. The Salesforce server URL (if blank, defaults to `www.salesforce.com`). To connect to a sandbox instance, change this to `test.salesforce.com`.
`pollWaitMillis`	Optional. Defaults to `10000`. The number of milliseconds to wait when polling for results of the deployment. Note that deployment may succeed even if you stop waiting.

Field	Description
checkOnly	Optional. Defaults to false. Set to true to check the validity of the deployed files without making any changes in the organization. This will not deploy any components or change the organization in any way.
maxPoll	Optional. Defaults to 20. The number of times to poll the server for the results of the deploy request. Note that deployment may succeed even if you stop waiting.
deployRoot	Required if zipFile is not specified. Specifies the root of the directory tree of files to deploy. You must define a value for either zipFile or deployRoot.
zipFile	Required if deployRoot is not specified.. Specifies the path of the metadata zip file to be deployed. You must define a value for either zipFile or deployRoot.
singlePackage	Optional. Defaults to false. Declares that the zipFile or deployRoot parameter points to a directory structure with a single package, as opposed to a set of packages.
allowMissingFiles	Optional. Defaults to false. Specifies whether a deploy succeeds even if files that are specified in package.xml are not in the zip file. Do not use this parameter for deployment to production organizations.
autoUpdatePackage	Optional. Defaults to false. Specifies whether a deploy should continue even if files present in the zip file are not specified in package.xml. Do not use this parameter for deployment to production organizations.
ignoreWarnings	Optional. Defaults to false. This setting indicates that a deployment should succeed even if there are warnings (true) or that one or more warnings will cause the deployment to fail and roll back (false). If there are errors, as opposed to warnings, the deployment will always fail and roll back.
rollbackOnError	Optional. Defaults to true. Indicates whether any failure causes a complete rollback (true) or not (false). If false, whatever set of actions can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to true if you are deploying to a production organization.
runAllTests	Optional. Defaults to false. If set to true, all tests are run after deploy. For deployment to a production organization, all tests, except for those that originate from installed managed packages, are automatically run regardless of this argument. If any of these tests fail when the rollbackOnError parameter is set to true, the deployment is rolled back and no changes will be made to your organization.
runTest	Optional child elements. A list of Apex classes containing tests run after deploy. If any of these tests fail when the rollbackOnError parameter is set to true, the deployment is rolled back and no changes will be made to your organization.
trace	Optional. Defaults to false. Prints the SOAP requests and responses to the console. Note that this will show the user's password in plain text during login.
logType	Optional. The debug logging level for running tests. The default is None. Valid options are: • None • Debugonly • Db

Field	Description
	• Profiling • Callout • Detail

Note: The Force.com Migration Tool ignores any files or folders with a name starting with a period (.) or ending with a tilde (~) when deploying files. Some source control systems, such as Subversion, create files or folders with names starting with a period. These files can cause issues during deployment to Salesforce, so the Force.com Migration Tool ignores them.

The Force.com Migration Tool comes with a sample `build.xml` file that lists several deployment targets. You will want to create your own custom targets using the sample targets as starting points. A description of the sample targets follows:

- `deployUnpackaged` — deploys unpackaged components specified in the target.
- `deployCode` — deploys the contents of the `codepkg` package specified in the target.
- `undeployCode` — deletes classes and triggers in the `removecodepkg` directory specified by the `destructiveChanges.xml` manifest. This file is similar to `package.xml`, but lists components to be deleted. For more information, see Deleting Files from an Organization on page 22.
- `deployCodeFailingTest` — deploys code that fails testing requirements, strictly for demonstration purposes.
- `deployCodeCheckOnly` — verifies that deployment will work, but does not deploy any components.

Deploying Components

You can deploy any set of components as a package or into your organization directly in the unpackaged package. The package used is not determined by the `build.xml` target, but by the project manifest (`package.xml`). A sample deployment target follows:

```
<target name="deployUnpackaged">
  <sf:deploy
      username="${sf.username}"
      password="${sf.password}"
      serverurl="${sf.serverurl}"
      deployroot="projectFolder"/>
</target>
```

Deploying Code

You can deploy metadata components and Apex at the same time, but you may find it useful to create separate targets for deploying Apex, so that you can run tests as part of the deployment. A portion of a `build.xml` file is listed below, with a target named `deployCode` that deploys the contents of the `codepkg` package and runs the tests for one class.

```
<target name="deployCode">
  <sf:deploy
      username="${sf.username}"
      password="${sf.password}"
      serverurl="${sf.serverurl}"
      deployroot="codepkg">
    <runTest>SampleDeployClass</runTest>
```

```
  </sf:deploy>
</target>
```

Running Tests in a Deployment

For deployment to a production organization, all the tests in your organization, except for those that originate from installed managed packages, are automatically run. If any of the tests fail, the entire deployment will roll back.

There is an exception to this rule if you are deploying components for one or more of the following metadata types:

- ApexComponent
- ApexPage
- Dashboard
- EmailTemplate
- Report
- Scontrol
- StaticResource

If your deployment consists entirely of components for one or more of these metadata types, no tests are run. However, if the deployment includes components for any other metadata type, all the tests are automatically run.

For example, no tests are run for the following deployments:

- One ApexComponent component
- 100 Report components and 40 Dashboard components

All tests are automatically run for the following deployments:

- One CustomField component
- One ApexComponent component and one ApexClass component
- Five CustomField components and one ApexPage component
- 100 Report components, 40 Dashboard components, and one CustomField component

Checking the Status of a Task

When you run a target, the Force.com Migration Tool outputs the request ID for each deploy or retrieve task included in the target. You can use the request ID to check the status of a deploy or retrieve task. This is particularly useful if a client is running for a long time and there is a network issue that breaks the connectivity between the Force.com Migration Tool on your machine and Salesforce.

To check the status of a run target, use the following command:

```
ant -Dsf.asyncRequestId=requestID targetName
```

Use the *requestID* that was printed out when you ran the target. For example, if you run the `deployZip` target, you can run the following command to check the status of the retrieval:

```
ant -Dsf.asyncRequestId=04sx00000002aGuAAI deployZip
```

Note: You should not use this command with targets that contain multiple retrieve or deploy tasks. You should not use this command with the `bulkRetrieve` task also as this task batches retrievals in multiple requests.

To track the status of deployments from within Salesforce, you can also from Setup, click **Monitor** > **Deployments**.

Chapter 4

Common Migration Issues

There are a number of common issues you may run into when migrating metadata and deploying changes. These issues can be grouped into three categories:

- Metadata — Special considerations for dealing with certain metadata components
- Connectivity — Problems connecting to an organization or polling for results
- Testing and Code Coverage — Testing Apex before deployment

Common Metadata Issues

The most common metadata issues are detailed below:

- Retrieving custom fields on standard objects — When you use the wildcard symbol in `package.xml`, to retrieve all objects, you will not retrieve standard objects, or custom fields on standard objects. To retrieve custom fields on standard objects, see Constructing a Project Manifest on page 6.
- Profiles or permission sets and field-level security — The contents of a retrieved profile or permission set depend on the other contents of the retrieve request. For example, field-level security for fields included in custom objects are returned at the same time as profiles or permission sets. For more information, see Profile and PermissionSet in the *Force.com Metadata API Developer's Guide*.
- Understanding packages — Packages are used to bundle related components so they can be shared with multiple organizations, or distributed on Force.com AppExchange. Managed packages are packages that can be upgraded in the installer's organization. They differ from unmanaged packages in that some components are locked, in order to permit upgrades. Metadata components that are not in any package can be accessed with the `unpackaged` attribute of `sf:retrieve` and `sf:deploy`.
- Workflow — A `.workflow` file is a container for the individual workflow components associated with an object, including WorkflowAlert, WorkflowFieldUpdate, WorkflowOutboundMessage, WorkflowRule, and WorkflowTask. To retrieve all workflows, include the following XML in `package.xml`:

```
<types>
    <members>*</members>
    <name>Workflow</name>
</types>
```

- Retrieving or deploying components that depend on an object definition — The following metadata components are dependent on a particular object for their definition: `CustomField`, `Picklist`, `RecordType`, `Weblink`, and `ValidationRule`. This means you must dot-qualify the component name with the object name in `package.xml`, and may not use the wildcard symbol. For more information, see Constructing a Project Manifest on page 6.
- Personal folders — Users' personal folders, for both reports and documents, are not exposed in the Metadata API. To migrate reports or documents you must move them to a public folder.

Connection Problems

The most common connection problems are detailed below:

- Request timed out — When you retrieve or deploy metadata, the request is sent asynchronously, meaning that the response is not returned immediately. Because these calls are asynchronous, the server will process a `deploy()` to completion even if the Force.com Migration Tool times out. If the deploy succeeds, the changes will be committed to the server. If the deploy fails after timing out, there is no way to retrieve the error message from the server. For this reason, it is important to tune your `pollWaitMillis` and `maxPoll` parameters if you receive time-out errors:
  - `pollWaitMillis` — The number of milliseconds to wait between polls for retrieve/deploy results. The default value is `10000` milliseconds (ten seconds). For long-running processes, increase this number to reduce the total number of polling requests, which count against your daily API limits.
  - `maxPoll` — The number of polling attempts to be performed before aborting. The default value is `20`. When combined with the default value of `pollWaitMillis (10000)`, this means the Force.com Migration Tool will give the server process a total of 200 seconds (20 poll attempts, waiting 10 seconds between them) to complete before timing out.

Note: Since these parameters have default values, they do not have to be specified, and may not exist on your named targets. To add these parameters, include them in the target definition. For example:

```
<sf:retrieve
  username="${sf.username}"
  password="${sf.password}"
  serverurl="${sf.serverurl}"
  retrieveTarget="retrieveUnpackaged"
. unpackaged="unpackaged/package.xml"
  pollWaitMillis="10000"
  maxPoll="100"/>
```

- Invalid username, password, security token; or user locked out - This error indicates a problem with the credentials in `build.properties`:
  - Username — Verify that your username is correct on this account. When connecting to a sandbox instance your sandbox name is appended to your username. For example, if your production username is `foo@salesforce.com`, and one of your sandboxes is called `bar`, then your sandbox username is `foo@salesforce.com.bar`.
  - Password — Verify that your password is correct for this account. Note that you security token is appended to the end of your pasword.
  - Security token — The security token is a 25-digit string appended to your password. To reset your security token, go to the Reset My Security Token page in your personal settings.
  - Locked out — If you unsuccessfully attempt to log into an organization too many times, you may be temporarily locked out. The number of times you may fail to connect and the lockout duration depend on your organization settings. Either contact your administrator to have the lock removed, or wait until the lockout period expires.
  - Connection mismatch between sandbox and production — If all of your connection credentials in `build.properties` are correct, you may have a URL mismatch. The server URL is different for sandbox and production environments. In `build.properties`, the `sf.serverurl` value for production is `https://login.salesforce.com`. For sandbox environments, the value is `https://test.salesforce.com`.
- Proxy settings — If you connect through a proxy, you will need to follow the Ant Proxy Configuration instructions.

Testing in Apex

When deploying to a production organization, every unit test in your organization namespace is executed. Before you deploy Apex, the following must be true:

- At least 75% of your Apex code must be covered by unit tests, and all of those tests must complete successfully.

  Note the following.
  - When deploying to a production organization, every unit test in your organization namespace is executed.
  - Calls to `System.debug` are not counted as part of Apex code coverage.

    - Test methods and test classes are not counted as part of Apex code coverage.
    - While only 75% of your Apex code must be covered by tests, your focus shouldn't be on the percentage of code that is covered. Instead, you should make sure that every use case of your application is covered, including positive and negative cases, as well as bulk and single record. This should lead to 75% or more of your code being covered by unit tests.
- Every trigger must have some test coverage.
- All classes and triggers must compile successfully.

Salesforce.com recommends that you write tests for the following:

Single action

Test to verify that a single record produces the correct, expected result.

Bulk actions

Any Apex code, whether a trigger, a class or an extension, may be invoked for 1 to 200 records. You must test not only the single record case, but the bulk cases as well.

Positive behavior

Test to verify that the expected behavior occurs through every expected permutation, that is, that the user filled out everything correctly and did not go past the limits.

Negative behavior

There are likely limits to your applications, such as not being able to add a future date, not being able to specify a negative amount, and so on. You must test for the negative case and verify that the error messages are correctly produced as well as for the positive, within the limits cases.

Restricted user

Test whether a user with restricted access to the sObjects used in your code sees the expected behavior. That is, whether they can run the code or receive error messages.

Note: Conditional and ternary operators are not considered executed unless both the positive and negative branches are executed.

For more information, see "Understanding Testing in Apex" in the *Force.com Apex Code Developer's Guide*.

Glossary

A |B |C |D |E |F |G |H |I |J |K |L |M |N |O |P |Q |R |S |T |U |V |W |X |Y |Z

A

Apex

Apex is a strongly typed, object-oriented programming language that allows developers to execute flow and transaction control statements on the Force.com platform server in conjunction with calls to the Force.com API. Using syntax that looks like Java and acts like database stored procedures, Apex enables developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages. Apex code can be initiated by Web service requests and from triggers on objects.

Apex-Managed Sharing

Enables developers to programmatically manipulate sharing to support their application's behavior. Apex-managed sharing is only available for custom objects.

App

Short for "application." A collection of components such as tabs, reports, dashboards, and Visualforce pages that address a specific business need. Salesforce provides standard apps such as Sales and Call Center. You can customize the standard apps to match the way you work. In addition, you can package an app and upload it to the AppExchange along with related components such as custom fields, custom tabs, and custom objects. Then, you can make the app available to other Salesforce users from the AppExchange.

AppExchange

The AppExchange is a sharing interface from salesforce.com that allows you to browse and share apps and services for the Force.com platform.

AppExchange Upgrades

Upgrading an app is the process of installing a newer version.

Application Programming Interface (API)

The interface that a computer system, library, or application provides to allow other computer programs to request services from it and exchange data.

Asynchronous Calls

A call that does not return results immediately because the operation may take a long time. Calls in the Metadata API and Bulk API are asynchronous.

B

Boolean Operators

You can use Boolean operators in report filters to specify the logical relationship between two values. For example, the AND operator between two values yields search results that include both values. Likewise, the OR operator between two values yields search results that include either value.

Bulk API

The REST-based Bulk API is optimized for processing large sets of data. It allows you to query, insert, update, upsert, or delete a large number of records asynchronously by submitting a number of batches which are processed in the background by Salesforce. See also SOAP API.

C

Class, Apex

A template or blueprint from which Apex objects are created. Classes consist of other classes, user-defined methods, variables, exception types, and static initialization code. In most cases, Apex classes are modeled on their counterparts in Java.

Client App

An app that runs outside the Salesforce user interface and uses only the Force.com API or Bulk API. It typically runs on a desktop or mobile device. These apps treat the platform as a data source, using the development model of whatever tool and platform for which they are designed.

Component, Metadata

A component is an instance of a metadata type in the Metadata API. For example, CustomObject is a metadata type for custom objects, and the `MyCustomObject__c` component is an instance of a custom object. A component is described in an XML file and it can be deployed or retrieved using the Metadata API, or tools built on top of it, such as the Force.com IDE or the Force.com Migration Tool.

Component, Visualforce

Something that can be added to a Visualforce page with a set of tags, for example, `<apex:detail>`. Visualforce includes a number of standard components, or you can create your own custom components.

Component Reference, Visualforce

A description of the standard and custom Visualforce components that are available in your organization. You can access the component library from the development footer of any Visualforce page or the *Visualforce Developer's Guide*.

Controller, Visualforce

An Apex class that provides a Visualforce page with the data and business logic it needs to run. Visualforce pages can use the standard controllers that come by default with every standard or custom object, or they can use custom controllers.

Controlling Field

Any standard or custom picklist or checkbox field whose values control the available values in one or more corresponding dependent fields.

Custom App

See App.

Custom Field

A field that can be added in addition to the standard fields to customize Salesforce for your organization's needs.

Custom Help

Custom text administrators create to provide users with on-screen information specific to a standard field, custom field, or custom object.

Custom Links

Custom links are URLs defined by administrators to integrate your Salesforce data with external websites and back-office systems. Formerly known as Web links.

Custom Object

Custom records that allow you to store information unique to your organization.

Custom S-Control

Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

Custom Web content for use in custom links. Custom s-controls can contain any type of content that you can display in a browser, for example a Java applet, an Active-X control, an Excel file, or a custom HTML Web form.

D

Database

An organized collection of information. The underlying architecture of the Force.com platform includes a database where your data is stored.

Database Table

A list of information, presented with rows and columns, about the person, thing, or concept you want to track. See also Object.

Data Manipulation Language (DML)

An Apex method or operation that inserts, updates, or deletes records from the Force.com platform database.

Decimal Places

Parameter for number, currency, and percent custom fields that indicates the total number of digits you can enter to the right of a decimal point, for example, 4.98 for an entry of 2. Note that the system rounds the decimal numbers you enter, if necessary. For example, if you enter 4.986 in a field with `Decimal Places` of 2, the number rounds to 4.99. Salesforce uses the round half-up rounding algorithm. Half-way values are always rounded up. For example, 1.45 is rounded to 1.5. –1.45 is rounded to –1.5.

Dependent Field

Any custom picklist or multi-select picklist field that displays available values based on the value selected in its corresponding controlling field.

Developer Edition

A free, fully-functional Salesforce organization designed for developers to extend, integrate, and develop with the Force.com platform. Developer Edition accounts are available on developer.force.com.

Developer Force

The Developer Force website at developer.force.com provides a full range of resources for platform developers, including sample code, toolkits, an online developer community, and the ability to obtain limited Force.com platform environments.

Document Library

A place to store documents without attaching them to accounts, contacts, opportunities, or other records.

E

Email Alert

Email alerts are workflow and approval actions that are generated using an email template by a workflow rule or approval process and sent to designated recipients, either Salesforce users or others.

Email Template

A form email that communicates a standard message, such as a welcome letter to new employees or an acknowledgement that a customer service request has been received. Email templates can be personalized with merge fields, and can be written in text, HTML, or custom format.

Enterprise Edition

A Salesforce edition designed for larger, more complex businesses.

Enterprise WSDL

A strongly-typed WSDL for customers who want to build an integration with their Salesforce organization only, or for partners who are using tools like Tibco or webMethods to build integrations that require strong typecasting. The downside of the Enterprise WSDL is that it only works with the schema of a single Salesforce organization because it is bound to all of the unique objects and fields that exist in that organization's data model.

Entity Relationship Diagram (ERD)

A data modeling tool that helps you organize your data into entities (or objects, as they are called in the Force.com platform) and define the relationships between them. ERD diagrams for key Salesforce objects are published in the *SOAP API Developer's Guide*.

Enumeration Field

An enumeration is the WSDL equivalent of a picklist field. The valid values of the field are restricted to a strict set of possible values, all having the same data type.

F

Field

A part of an object that holds a specific piece of information, such as a text or currency value.

Field-Level Security

Settings that determine whether fields are hidden, visible, read only, or editable for users. Available in Enterprise, Unlimited, and Developer Editions only.

Filter Condition/Criteria

Condition on particular fields that qualifies items to be included in a list view or report, such as “State equals California.”

Force.com

The salesforce.com platform for building applications in the cloud. Force.com combines a powerful user interface, operating system, and database to allow you to customize and deploy applications in the cloud for your entire enterprise.

Force.com IDE

An Eclipse plug-in that allows developers to manage, author, debug and deploy Force.com applications in the Eclipse development environment.

Force.com Migration Tool

A toolkit that allows you to write an Apache Ant build script for migrating Force.com components between a local file system and a Salesforce organization.

Foreign Key

A field whose value is the same as the primary key of another table. You can think of a foreign key as a copy of a primary key from another table. A relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Formula Field

A type of custom field. Formula fields automatically calculate their values based on the values of merge fields, expressions, or other values.

Function

Built-in formulas that you can customize with input parameters. For example, the DATE function creates a date field type from a given year, month, and day.

G

Gregorian Year

A calendar based on a 12-month structure used throughout much of the world.

H

HTTP Debugger

An application that can be used to identify and inspect SOAP requests that are sent from the AJAX Toolkit. They behave as proxy servers running on your local machine and allow you to inspect and author individual requests.

I

ID

See Salesforce Record ID.

Inline S-Control

Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

An s-control that displays within a record detail page or dashboard, rather than on its own page.

Instance

The cluster of software and hardware represented as a single logical server that hosts an organization's data and runs their applications. The Force.com platform runs on multiple instances, but data for any single organization is always consolidated on a single instance.

Integration User

A Salesforce user defined solely for client apps or integrations. Also referred to as the logged-in user in a SOAP API context.

ISO Code

The International Organization for Standardization country code, which represents each country by two letters.

J

Junction Object

A custom object with two master-detail relationships. Using a custom junction object, you can model a "many-to-many" relationship between two objects. For example, you may have a custom object called "Bug" that relates to the standard case object such that a bug could be related to multiple cases and a case could also be related to multiple bugs.

K

No Glossary items for this entry.

L

License Management Application (LMA)

A free AppExchange app that allows you to track sales leads and accounts for every user who downloads your managed package (app) from the AppExchange.

License Management Organization (LMO)

The Salesforce organization that you use to track all the Salesforce users who install your package. A license management organization must have the License Management Application (LMA) installed. It automatically receives notification every time your package is installed or uninstalled so that you can easily notify users of upgrades. You can specify any Enterprise, Unlimited, or Developer Edition organization as your license management organization. For more information, go to http://www.salesforce.com/docs/en/lma/index.htm.

List View

A list display of items (for example, accounts or contacts) based on specific criteria. Salesforce provides some predefined views.

In the Console tab, the list view is the top frame that displays a list view of records based on specific criteria. The list views you can select to display in the console are the same list views defined on the tabs of other objects. You cannot create a list view within the console.

Local Project

A `.zip` file containing a project manifest (`package.xml` file) and one or more metadata components.

Locale

The country or geographic region in which the user is located. The setting affects the format of date and number fields, for example, dates in the English (United States) locale display as 06/30/2000 and as 30/06/2000 in the English (United Kingdom) locale.

In Professional, Enterprise, Unlimited, and Developer Edition organizations, a user's individual `Locale` setting overrides the organization's `Default Locale` setting. In Personal and Group Editions, the organization-level locale field is called `Locale`, not `Default Locale`.

Logged-in User

In a SOAP API context, the username used to log into Salesforce. Client applications run with the permissions and sharing of the logged-in user. Also referred to as an integration user.

Lookup Field

A type of field that contains a linkable value to another record. You can display lookup fields on page layouts where the object has a lookup or master-detail relationship with another object. For example, cases have a lookup relationship with assets that allows users to select an asset using a lookup dialog from the case edit page and click the name of the asset from the case detail page.

M

Managed Package

A collection of application components that is posted as a unit on the AppExchange and associated with a namespace and possibly a License Management Organization. To support upgrades, a package must be managed. An organization can create a single managed package that can be downloaded and installed by many different organizations. Managed packages differ from unmanaged packages by having some locked components, allowing the managed package to be upgraded later. Unmanaged packages do not include locked components and cannot be upgraded. In addition, managed packages obfuscate certain components (like Apex) on subscribing organizations to protect the intellectual property of the developer.

Manifest File

The project manifest file (`package.xml`) lists the XML components to retrieve or deploy when working with the Metadata API, or clients built on top of the Metadata API, such as the Force.com IDE or the Force.com Migration Tool.

Manual Sharing

Record-level access rules that allow record owners to give read and edit permissions to other users who might not have access to the record any other way.

Many-to-Many Relationship

A relationship where each side of the relationship can have many children on the other side. Many-to-many relationships are implemented through the use of junction objects.

Master-Detail Relationship

A relationship between two different types of records that associates the records with each other. For example, accounts have a master-detail relationship with opportunities. This type of relationship affects record deletion, security, and makes the lookup relationship field required on the page layout.

Metadata

Information about the structure, appearance, and functionality of an organization and any of its parts. Force.com uses XML to describe metadata.

Metadata WSDL

A WSDL for users who want to use the Force.com Metadata API calls.

Multitenancy

An application model where all users and apps share a single, common infrastructure and code base.

N

Namespace

In a packaging context, a one- to 15-character alphanumeric identifier that distinguishes your package and its contents from packages of other developers onAppExchange, similar to a domain name. Salesforce automatically prepends your namespace prefix, followed by two underscores ("__"), to all unique component names in your Salesforce organization.

Native App

An app that is built exclusively with setup (metadata) configuration on Force.com. Native apps do not require any external services or infrastructure.

O

Object

An object allows you to store information in your Salesforce organization. The object is the overall definition of the type of information you are storing. For example, the case object allow you to store information regarding customer inquiries.

For each object, your organization will have multiple records that store the information about specific instances of that type of data. For example, you might have a case record to store the information about Joe Smith's training inquiry and another case record to store the information about Mary Johnson's configuration issue.

Object-Level Help

Custom help text that you can provide for any custom object. It displays on custom object record home (overview), detail, and edit pages, as well as list views and related lists.

Object-Level Security

Settings that allow an administrator to hide whole objects from users so that they don't know that type of data exists. Object-level security is specified with object permissions.

onClick JavaScript

JavaScript code that executes when a button or link is clicked.

One-to-Many Relationship

A relationship in which a single object is related to many other objects. For example, an account may have one or more related contacts.

Organization-Wide Defaults

Settings that allow you to specify the baseline level of data access that a user has in your organization. For example, you can set organization-wide defaults so that any user can see any record of a particular object that is enabled via their object permissions, but they need extra permissions to edit one.

Outbound Message

An outbound message is a workflow, approval, or milestone action that sends the information you specify to an endpoint you designate, such as an external service. An outbound message sends the data in the specified fields in the form of a SOAP message to the endpoint. Outbound messaging is configured in the Salesforce setup menu. Then you must configure the external endpoint. You can create a listener for the messages using the SOAP API.

Overlay

An overlay displays additional information when you hover your mouse over certain user interface elements. Depending on the overlay, it will close when you move your mouse away, click outside of the overlay, or click a close button.

Owner

Individual user to which a record (for example, a contact or case) is assigned.

P

Package

A group of Force.com components and applications that are made available to other organizations through the AppExchange. You use packages to bundle an app along with any related components so that you can upload them to AppExchange together.

Partner WSDL

A loosely-typed WSDL for customers, partners, and ISVs who want to build an integration or an AppExchange app that can work across multiple Salesforce organizations. With this WSDL, the developer is responsible for marshaling data in the correct object representation, which typically involves editing the XML. However, the developer is also freed from being dependent on any particular data model or Salesforce organization. Contrast this with the Enterprise WSDL, which is strongly typed.

Picklist

Selection list of options available for specific fields in a Salesforce object, for example, the `Industry` field for accounts. Users can choose a single value from a list of options rather than make an entry directly in the field. See also Master Picklist.

Picklist (Multi-Select)

Selection list of options available for specific fields in a Salesforce object. Multi-select picklists allow users to choose one or more values. Users can choose a value by double clicking on it, or choose additional values from a scrolling list by holding down the CTRL key while clicking a value and using the arrow icon to move them to the selected box.

Picklist Values

Selections displayed in drop-down lists for particular fields. Some values come predefined, and other values can be changed or defined by an administrator.

Primary Key

A relational database concept. Each table in a relational database has a field in which the data value uniquely identifies the record. This field is called the primary key. The relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Production Organization

A Salesforce organization that has live users accessing data.

Professional Edition

A Salesforce edition designed for businesses who need full-featured CRM functionality.

Q

Queue

A holding area for items before they are processed. Salesforce uses queues in a number of different features and technologies.

Query String Parameter

A name-value pair that's included in a URL, typically after a '?' character. For example:

```
http://na1.salesforce.com/001/e?name=value
```

R

Record

A single instance of a Salesforce object. For example, “John Jones” might be the name of a contact record.

Record Name

A standard field on all Salesforce objects. Whenever a record name is displayed in a Force.com application, the value is represented as a link to a detail view of the record. A record name can be either free-form text or an autonumber field. `Record Name` does not have to be a unique value.

Record Type

A record type is a field available for certain records that can include some or all of the standard and custom picklist values for that record. You can associate record types with profiles to make only the included picklist values available to users with that profile.

Record-Level Security

A method of controlling data in which you can allow a particular user to view and edit an object, but then restrict the records that the user is allowed to see.

Recycle Bin

A page that lets you view and restore deleted information. Access the Recycle Bin by using the link in the sidebar.

Related Object

Objects chosen by an administrator to display in the Console tab's mini view when records of a particular type are shown in the console's detail view. For example, when a case is in the detail view, an administrator can choose to display an associated account, contact, or asset in the mini view.

Relationship

A connection between two objects, used to create related lists in page layouts and detail levels in reports. Matching values in a specified field in both objects are used to link related data; for example, if one object stores data about companies and another object stores data about people, a relationship allows you to find out which people work at the company.

Relationship Query

In a SOQL context, a query that traverses the relationships between objects to identify and return results. Parent-to-child and child-to-parent syntax differs in SOQL queries.

Report Type

A *report type* defines the set of records and fields available to a report based on the relationships between a primary object and its related objects. Reports display only records that meet the criteria defined in the report type. Salesforce provides a set of pre-defined standard report types; administrators can create custom report types as well.

Role Hierarchy

A record-level security setting that defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.

Roll-Up Summary Field

A field type that automatically provides aggregate values from child records in a master-detail relationship.

S

SaaS

See Software as a Service (SaaS).

S-Control

Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

Custom Web content for use in custom links. Custom s-controls can contain any type of content that you can display in a browser, for example a Java applet, an Active-X control, an Excel file, or a custom HTML Web form.

Salesforce SOA (Service-Oriented Architecture)

A powerful capability of Force.com that allows you to make calls to external Web services from within Apex.

Sandbox Organization

A nearly identical copy of a Salesforce production organization. You can create multiple sandboxes in separate environments for a variety of purposes, such as testing and training, without compromising the data and applications in your production environment.

Search Layout

The organization of fields included in search results, in lookup dialogs, and in the key lists on tab home pages.

Session ID

An authentication token that is returned when a user successfully logs in to Salesforce. The Session ID prevents a user from having to log in again every time he or she wants to perform another action in Salesforce. Different from a record ID or Salesforce ID, which are terms for the unique ID of a Salesforce record.

Session Timeout

The period of time after login before a user is automatically logged out. Sessions expire automatically after a predetermined length of inactivity, which can be configured in Salesforce from Setup by clicking **Security Controls**. The default is 120 minutes (two hours). The inactivity timer is reset to zero if a user takes an action in the Web interface or makes an API call.

Setup

A menu where administrators can customize and define organization settings and Force.com apps. Depending on your organization's user interface settings, Setup may be a link in the user interface header or in the drop-down list under your name.

Sharing

Allowing other users to view or edit information you own. There are different ways to share data:

- Sharing Model—defines the default organization-wide access levels that users have to each other's information and whether to use the hierarchies when determining access to data.
- Role Hierarchy—defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.
- Sharing Rules—allow an administrator to specify that all information created by users within a given group or role is automatically shared to the members of another group or role.
- Manual Sharing—allows individual users to share records with other users or groups.
- Apex-Managed Sharing—enables developers to programmatically manipulate sharing to support their application's behavior. See Apex-Managed Sharing.

Sharing Model

Behavior defined by your administrator that determines default access by users to different types of records.

Sharing Rule

Type of default sharing created by administrators. Allows users in a specified group or role to have access to all information created by users within a given group or role.

Sites

Force.com Sites enables you to create public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password.

Snippet

Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

A type of s-control that is designed to be included in other s-controls. Similar to a helper method that is used by other methods in a piece of code, a snippet allows you to maintain a single copy of HTML or JavaScript that you can reuse in multiple s-controls.

SOAP (Simple Object Access Protocol)

A protocol that defines a uniform way of passing XML-encoded data.

Software as a Service (SaaS)

A delivery model where a software application is hosted as a service and provided to customers via the Internet. The SaaS vendor takes responsibility for the daily maintenance, operation, and support of the application and each customer's data. The service alleviates the need for customers to install, configure, and maintain applications with their own hardware, software, and related IT resources. Services can be delivered using the SaaS model to any market segment.

SOQL (Salesforce Object Query Language)

A query language that allows you to construct simple but powerful query strings and to specify the criteria that should be used to select data from the Force.com database.

SOSL (Salesforce Object Search Language)

A query language that allows you to perform text-based searches using the Force.com API.

Standard Object

A built-in object included with the Force.com platform. You can also build custom objects to store information that is unique to your app.

System Log

Part of the Developer Console, a separate window console that can be used for debugging code snippets. Enter the code you want to test at the bottom of the window and click Execute. The body of the System Log displays system resource information, such as how long a line took to execute or how many database calls were made. If the code did not run to completion, the console also displays debugging information.

T

Test Method

An Apex class method that verifies whether a particular piece of code is working properly. Test methods take no arguments, commit no data to the database, and can be executed by the `runTests()` system method either through the command line or in an Apex IDE, such as the Force.com IDE.

Translation Workbench

The Translation Workbench lets you specify languages you want to translate, assign translators to languages, create translations for customizations you've made to your Salesforce organization, and override labels and translations from managed packages. Everything from custom picklist values to custom fields can be translated so your global users can use all of Salesforce in their language.

Trigger

A piece of Apex that executes before or after records of a particular type are inserted, updated, or deleted from the database. Every trigger runs with a set of context variables that provide access to the records that caused the trigger to fire, and all triggers run in bulk mode—that is, they process several records at once, rather than just one record at a time.

Trigger Context Variable

Default variables that provide access to information about the trigger and the records that caused it to fire.

U

Unit Test

A unit is the smallest testable part of an application, usually a method. A unit test operates on that piece of code to make sure it works correctly. See also Test Method.

Unlimited Edition

Unlimited Edition is salesforce.com's flagship solution for maximizing CRM success and extending that success across the entire enterprise through the Force.com platform.

Unmanaged Package

A package that cannot be upgraded or controlled by its developer.

URL (Uniform Resource Locator)

The global address of a website, document, or other resource on the Internet. For example, http://www.salesforce.com.

URL S-Control

Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

An s-control that contains an external URL that hosts the HTML that should be rendered on a page. When saved this way, the HTML is hosted and run by an external website. URL s-controls are also called Web controls.

V

Validation Rule

A rule that prevents a record from being saved if it does not meet the standards that are specified.

Visualforce

A simple, tag-based markup language that allows developers to easily define custom pages and components for apps built on the platform. Each tag corresponds to a coarse or fine-grained component, such as a section of a page, a related list, or a field. The components can either be controlled by the same logic that is used in standard Salesforce pages, or developers can associate their own logic with a controller written in Apex.

W

Web Control

See URL S-Control.

Web Links

See Custom Links.

Web Service

A mechanism by which two applications can easily exchange data over the Internet, even if they run on different platforms, are written in different languages, or are geographically remote from each other.

WebService Method

An Apex class method or variable that can be used by external systems, like a mash-up with a third-party application. Web service methods must be defined in a global class.

Web Services API

A Web services application programming interface that provides access to your Salesforce organization's information. See also SOAP API and Bulk API.

Web Tab

A custom tab that allows your users to use external websites from within the application.

Workflow Action

A workflow action is an email alert, field update, outbound message, or task that fires when the conditions of a workflow rule are met.

Workflow Email Alert

A workflow action that sends an email when a workflow rule is triggered. Unlike workflow tasks, which can only be assigned to application users, workflow alerts can be sent to any user or contact, as long as they have a valid email address.

Workflow Field Update

A workflow action that changes the value of a particular field on a record when a workflow rule is triggered.

Workflow Outbound Message

A workflow action that sends data to an external Web service, such as another cloud computing application. Outbound messages are used primarily with composite apps.

Workflow Queue

A list of workflow actions that are scheduled to fire based on workflow rules that have one or more time-dependent workflow actions.

Workflow Rule

A workflow rule sets workflow actions into motion when its designated conditions are met. You can configure workflow actions to execute immediately when a record meets the conditions in your workflow rule, or set time triggers that execute the workflow actions on a specific day.

Workflow Task

A workflow action that assigns a task to an application user when a workflow rule is triggered.

WSDL (Web Services Description Language) File

An XML file that describes the format of messages you send and receive from a Web service. Your development environment's SOAP client uses the Salesforce Enterprise WSDL or Partner WSDL to communicate with Salesforce using the SOAP API.

X

XML (Extensible Markup Language)

A markup language that enables the sharing and transportation of structured data. All Force.com components that are retrieved or deployed through the Metadata API are represented by XML definitions.

Y

No Glossary items for this entry.

Z

Zip File

A data compression and archive format.

A collection of files retrieved or deployed by the Metadata API. See also Local Project.

Index

S

T

U

V

W