



Version 30.0: Spring '14

Database.com Metadata API Developer's Guide



Last updated: May 17, 2014

Table of Contents

GETTING STARTED.....	1
Chapter 1: Understanding Metadata API.....	1
Development Platforms.....	3
Standards Compliance.....	3
Metadata API Support Policy.....	3
Related Resources.....	4
Chapter 2: Quick Start.....	5
Prerequisites.....	5
Step 1: Generate or Obtain the Web Service WSDLs for Your Organization.....	5
Step 2: Import the WSDL Files Into Your Development Platform.....	6
Step 3: Walk Through the Java Sample Code.....	7
USING METADATA API.....	14
Chapter 3: Deploying and Retrieving Metadata.....	14
Working with the Zip File.....	14
Sample package.xml Manifest Files.....	16
Running Tests in a Deployment.....	19
Maintaining User References.....	20
Chapter 4: CRUD-Based Metadata Development.....	21
Chapter 5: Error Handling.....	24
Error Handling for Session Expiration	24
REFERENCE.....	25
Chapter 6: File-Based Calls.....	25
deploy().....	25
checkDeployStatus().....	31
cancelDeploy().....	32
retrieve().....	34
RetrieveRequest.....	39
checkRetrieveStatus().....	40
Chapter 7: CRUD-Based Calls.....	42
create().....	42
delete().....	43
update().....	44
createMetadata().....	46
readMetadata().....	48
updateMetadata().....	50
deleteMetadata().....	51

renameMetadata().....	53
Chapter 8: Utility Calls.....	55
checkStatus().....	55
describeMetadata().....	56
listMetadata().....	56
ListMetadataQuery.....	58
Chapter 9: Result Objects.....	59
AsyncResult.....	59
CancelDeployResult.....	61
DeployResult.....	62
DescribeMetadataResult.....	68
ReadResult.....	69
RetrieveResult.....	69
SaveResult.....	71
DeleteResult.....	71
Error.....	72
Chapter 10: Metadata Types.....	73
Metadata Components and Types.....	75
Unsupported Metadata Types.....	76
ApexClass.....	76
ApexTrigger.....	78
AppMenu.....	79
CallCenter.....	81
CustomObject.....	83
CustomField.....	87
NamedFilter.....	94
Picklist (Including Dependent Picklist).....	96
SharingReason.....	101
SharingRecalculation.....	102
ValidationRule.....	103
Weblink.....	104
Metadata Field Types.....	107
ExternalDataSource.....	109
Group.....	111
InstalledPackage.....	112
Layout.....	112
Metadata.....	124
MetadataWithContent.....	124
Package.....	125
PermissionSet.....	126
Profile.....	130
Queue.....	137

QuickAction.....	139
RemoteSiteSetting.....	142
Role.....	143
SamSsoConfig.....	144
Settings.....	146
AccountSettings.....	147
ActivitiesSettings.....	148
BusinessHoursSettings.....	151
LiveAgentSettings.....	155
MobileSettings.....	155
SecuritySettings.....	158
SharedTo.....	162
SharingRules.....	163
BaseSharingRule.....	165
CriteriaBasedSharingRule.....	166
OwnerSharingRule.....	168
SiteDotCom.....	169
Workflow.....	170
Glossary.....	179
Index.....	189

Understanding Metadata API

In this chapter ...

- [Development Platforms](#)
- [Standards Compliance](#)
- [Metadata API Support Policy](#)
- [Related Resources](#)

Use Metadata API to retrieve, deploy, create, update or delete customization information, such as custom object definitions and page layouts, for your organization. This API is intended for managing customizations and for building tools that can manage the metadata model, not the data itself. To create, retrieve, update or delete records, such as accounts or leads, use data [SOAP API](#) or [REST API](#).

The easiest way to access the functionality in Metadata API is to use the Force.com IDE or Force.com Migration Tool. These tools are built on top of Metadata API and use the standard Eclipse and Ant tools respectively to simplify the task of working with Metadata API. Built on the Eclipse platform, the Force.com IDE provides a comfortable environment for programmers familiar with integrated development environments, allowing you to code, compile, test, and deploy all from within the IDE itself. The Force.com Migration Tool is ideal if you want to use a script or a command-line utility for moving metadata between a local directory and a Database.com organization. For more information about the Force.com IDE or Force.com Migration Tool, see [developer.force.com](#).

The underlying calls of Metadata API have been exposed for you to use directly, if you prefer to build your own client applications. This guide gives you more information about working directly with Metadata API.

You can use the Metadata API to manage setup and customization information (metadata) for your organizations. For example:

- Export the customizations in your organization as XML metadata files. See [Working with the Zip File](#) and [retrieve\(\)](#).
- Migrate configuration changes between organizations. See [deploy\(\)](#) and [retrieve\(\)](#).
- Modify existing customizations in your organization using XML metadata files. See [deploy\(\)](#) and [retrieve\(\)](#).
- Manage customizations in your organization programmatically. See [CRUD-Based Metadata Development](#).

You can modify metadata in your test database. You can also create scripts to populate a new organization with your custom objects, custom fields, and other components.

See Also:

[Deploying and Retrieving Metadata](#)
[CRUD-Based Metadata Development](#)
[Metadata Components and Types](#)

Development Platforms

Metadata API supports both file-based and CRUD-based development.

File-Based Development

The declarative or file-based asynchronous Metadata API `deploy()` and `retrieve()` calls deploy or retrieve a .zip file that holds components in a set of folders, and a manifest file named `package.xml`. For more information, see [Deploying and Retrieving Metadata](#) on page 14. The easiest way to access the file-based functionality is to use the Force.com IDE or Force.com Migration Tool.

CRUD-Based Development

The [CRUD-based Metadata API calls](#) act upon the metadata components in a manner similar to the way synchronous API calls in the *enterprise WSDL* act upon objects. For more information about the enterprise WSDL, see the *SOAP API Developer's Guide*.

Use the asynchronous `create()`, `update()`, and `delete()` calls with the utility call `checkStatus()`. For more information, see [CRUD-Based Metadata Development](#).

Alternatively, you can use the synchronous equivalents of these calls that are listed in [CRUD-based Metadata API calls](#), such as `createMetadata()`.

Standards Compliance

Metadata API is implemented to comply with the following specifications:

Standard Name	Website
Simple Object Access Protocol (SOAP) 1.1	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
Web Service Description Language (WSDL) 1.1	http://www.w3.org/TR/2001/NOTE-wsdl-20010315
WS-I Basic Profile 1.1	http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

Metadata API Support Policy

Salesforce.com supports previous versions of Metadata API. However, your new client applications should use the most recent version of the Force.com Metadata API WSDL file to fully exploit the benefits of richer features and greater efficiency.

Backward Compatibility

Salesforce.com strives to make backward compatibility easy when using the Force.com platform.

Each new Database.com release consists of two components:

- A new release of platform software that resides on salesforce.com systems
- A new version of the API

For example, the Spring '07 release included API version 9.0 and the Summer '07 release included API version 10.0.

We maintain support for each API version across releases of the platform software. The API is backward compatible in that an application created to work with a given API version will continue to work with that same API version in future platform software releases.

Salesforce.com does not guarantee that an application written against one API version will work with future API versions: Changes in method signatures and data representations are often required as we continue to enhance the API. However, we strive to keep the API consistent from version to version with minimal, if any, changes required to port applications to newer API versions.

For example, an application written using API version 9.0, which shipped with the Spring '07 release, will continue to work with API version 9.0 on the Summer '07 release, and on future releases beyond that. However, that same application might not work with API version 10.0 without modifications to the application.

API End-of-Life

Salesforce.com is committed to supporting each API version for a minimum of three years from the date of first release. In order to mature and improve the quality and performance of the API, versions that are more than three years old might cease to be supported.

When an API version is to be deprecated, advance notice is given at least one year before support ends. Salesforce.com will directly notify customers using API versions planned for deprecation.

Related Resources

The salesforce.com developer website provides a full suite of developer toolkits, sample code, sample SOAP messages, community-based support, and other resources to help you with your development projects. Be sure to visit https://wiki.developerforce.com/index.php/Getting_Started for more information, or visit <http://developer.force.com/join> to sign up for a free Developer Edition account.

You can visit these websites to find out more about Database.com applications:

- [Developer Force](#) provides a wealth of information for developers.
- [Salesforce.com](#) for information about the Database.com application.
- [Salesforce.com Community](#) for services to ensure Database.com customer success.

Chapter 2

Quick Start

The easiest way to access the functionality in Metadata API is to use the Force.com IDE or Force.com Migration Tool. These tools are built on top of Metadata API and use the standard Eclipse and Ant tools respectively to simplify the task of working with Metadata API. Built on the Eclipse platform, the Force.com IDE provides a comfortable environment for programmers familiar with integrated development environments, allowing you to code, compile, test, and deploy all from within the IDE itself. The Force.com Migration Tool is ideal if you want to use a script or a command-line utility for moving metadata between a local directory and a Database.com organization. For more information about the Force.com IDE or Force.com Migration Tool, see developer.force.com.

However, the underlying calls of Metadata API have been exposed for you to use directly, if you prefer to build your own client applications. This quick start gives you all the information you need to start writing applications that directly use Metadata API to manage customizations for your organization. It shows you how to get started with File-Based Development. For an example of CRUD-Based Development, see [Java Sample Code for CRUD-Based Development](#).

Prerequisites

Make sure you complete these prerequisites before you start using Metadata API.

- Create a development environment.

It is strongly recommended that you use a test database for development.

- Identify a user that has the “API Enabled” and “Modify All Data” permissions. These permissions are required to access Metadata API calls.
- Install a SOAP client. Metadata API works with current SOAP development environments, including, but not limited to, Visual Studio® .NET and the Force.com Web Service Connector (WSC).

In this document, we provide Java examples based on WSC and JDK 6 (Java Platform Standard Edition Development Kit 6). To run the samples, first download the latest force-wsc JAR file and its dependencies (dependencies are listed on the page when you select a version) from mvnrepository.com/artifact/com.force.api/force-wsc/.



Note: Development platforms vary in their SOAP implementations. Implementation differences in certain development platforms might prevent access to some or all of the features in Metadata API.

Step 1: Generate or Obtain the Web Service WSDLs for Your Organization

To access Metadata API calls, you need a Web Service Description Language (WSDL) file. The WSDL file defines the Web service that is available to you. Your development platform uses this WSDL to generate stub code to access the Web service it defines. You can either obtain the WSDL file from your organization’s Database.com administrator, or you can generate it yourself if you have access to the WSDL download page in the Database.com user interface. For more information about WSDL, see <http://www.w3.org/TR/wsdl>.

Before you can access Metadata API calls, you must authenticate to use the Web service using the `login()` call, which is defined in the enterprise WSDL and the partner WSDL. Therefore, you must also obtain one of these WSDLs.

Any user with the “Modify All Data” permission can download the WSDL file to integrate and extend the Database.com platform. (The System Administrator profile has this permission.)

The sample code in [Step 3: Walk Through the Java Sample Code](#) on page 7 uses the enterprise WSDL, though the partner WSDL works equally well.

To generate the metadata and enterprise WSDL files for your organization:

1. Log in to your Database.com account. You must log in as an administrator or as a user who has the “Modify All Data” permission.
2. From Setup, click **Develop > API**.
3. Click **Generate Metadata WSDL** and save the XML WSDL file to your file system.
4. Click **Generate Enterprise WSDL** and save the XML WSDL file to your file system.

Step 2: Import the WSDL Files Into Your Development Platform

Once you have the WSDL files, import them into your development platform so that your development environment can generate the necessary objects for use in building client Web service applications. This section provides sample instructions for WSC. For instructions about other development platforms, see your platform’s product documentation.



Note: The process for importing WSDL files is identical for the metadata and enterprise WSDL files.

Instructions for Java Environments (WSC)

Java environments access the API through Java objects that serve as proxies for their server-side counterparts. Before using the API, you must first generate these objects from your organization’s WSDL file.

Each SOAP client has its own tool for this process. For WSC, use the `wsdlc` utility.



Note: Before you run `wsdlc`, you must have the WSC JAR file installed on your system and referenced in your classpath. You can download the latest `force-wsc` JAR file and its dependencies (dependencies are listed on the page when you select a version) from mvnrepository.com/artifact/com.force.api/force-wsc/.

The basic syntax for `wsdlc` is:

```
java -classpath pathToWsc;pathToWscDependencies com.sforce.ws.tools.wsdlc
pathToWsdL/WsdLFilename pathToOutputJar/OutputJarFilename
```

For example, on Windows:

```
java -classpath force-wsc-30.0.0.jar;ST4-4.0.7.jar;antlr-runtime-3.5.jar
com.sforce.ws.tools.wsdlc metadata.wsdl metadata.jar
```

On Mac OS X and Unix, use a colon instead of a semicolon in between items in the classpath:

```
java -classpath force-wsc-30.0.0.jar:ST4-4.0.7.jar:antlr-runtime-3.5.jar
com.sforce.ws.tools.wsdlc metadata.wsdl metadata.jar
```

`wsdlc` generates a JAR file and Java source code and bytecode files for use in creating client applications. Repeat this process for the enterprise WSDL to create an enterprise JAR file.

Step 3: Walk Through the Java Sample Code

Once you have imported the WSDL files, you can begin building client applications that use Metadata API. The sample is a good starting point for writing your own code.

Before you run the sample, modify your project and the code to:

1. Include the WSC JAR, its dependencies, and the JAR files you generated from the WSDLs.



Note: Although WSC has other dependencies, the following sample only requires Rhino (`js-1.7R2.jar`), which you can download from mvnrepository.com/artifact/rhino/js.

2. Update `USERNAME` and `PASSWORD` variables in the `MetadataLoginUtil.login()` method with your user name and password. If your current IP address isn't in your organization's trusted IP range, you'll need to append a security token to the password.
3. If you are using a test database, be sure to change the login URL.

Login Utility

Java users can use `ConnectorConfig` to connect to Enterprise, Partner, and Metadata SOAP API. `MetadataLoginUtil` creates a `ConnectorConfig` object and logs in using the Enterprise WSDL login method. Then it retrieves `sessionId` and `metadataServerUrl` to create a `ConnectorConfig` and connects to Metadata API endpoint. `ConnectorConfig` is defined in WSC.

The `MetadataLoginUtil` class abstracts the login code from the other parts of the sample, allowing portions of this code to be reused without change across different Database.com APIs.

```
import com.sforce.soap.enterprise.EnterpriseConnection;
import com.sforce.soap.enterprise.LoginResult;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;

/**
 * Login utility.
 */
public class MetadataLoginUtil {

    public static MetadataConnection login() throws ConnectionException {
        final String USERNAME = "user@company.com";
        // This is only a sample. Hard coding passwords in source files is a bad practice.
        final String PASSWORD = "password";
        final String URL = "https://login.salesforce.com/services/Soap/c/30.0";
        final LoginResult loginResult = loginToSalesforce(USERNAME, PASSWORD, URL);
        return createMetadataConnection(loginResult);
    }

    private static MetadataConnection createMetadataConnection(
        final LoginResult loginResult) throws ConnectionException {
        final ConnectorConfig config = new ConnectorConfig();
        config.setServiceEndpoint(loginResult.getMetadataServerUrl());
        config.setSessionId(loginResult.getSessionId());
        return new MetadataConnection(config);
    }

    private static LoginResult loginToSalesforce(
        final String username,
        final String password,
        final String loginUrl) throws ConnectionException {
        final ConnectorConfig config = new ConnectorConfig();
        config.setAuthEndpoint(loginUrl);
        config.setServiceEndpoint(loginUrl);
    }
}
```

```

        config.setManualLogin(true);
        return (new EnterpriseConnection(config)).login(username, password);
    }
}

```

Java Sample Code for File-Based Development

The sample code logs in using the [login utility](#). Then it displays a menu with retrieve, deploy, and exit.

The `retrieve()` and `deploy()` calls both operate on a `.zip` file named `components.zip`. The `retrieve()` call retrieves components from your organization into `components.zip`, and the `deploy()` call deploys the components in `components.zip` to your organization. If you save the sample to your computer and execute it, run the retrieve option first so that you have a `components.zip` file that you can subsequently deploy. After retrieve or deploy calls, it checks `checkStatus()` in a loop until the status value in `AsyncResult` indicates that the operation has completed.

The `retrieve()` call uses a manifest file to determine the components to retrieve from your organization. A sample `package.xml` manifest file follows. For more details on the manifest file structure, see [Working with the Zip File](#). For this sample, the manifest file retrieves all custom objects, custom tabs, and page layouts.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>*</members>
    <name>CustomObject</name>
  </types>
  <types>
    <members>*</members>
    <name>CustomTab</name>
  </types>
  <types>
    <members>*</members>
    <name>Layout</name>
  </types>
  <version>30.0</version>
</Package>

```

Note the error handling code that follows each API call.

```

import java.io.*;
import java.util.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import com.sforce.soap.metadata.*;

/**
 * Sample that logs in and shows a menu of retrieve and deploy metadata options.
 */
public class FileBasedDeployAndRetrieve {

    private MetadataConnection metadataConnection;

    private static final String ZIP_FILE = "components.zip";

    // manifest file that controls which components get retrieved
    private static final String MANIFEST_FILE = "package.xml";

    private static final double API_VERSION = 29.0;

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;

    // maximum number of attempts to deploy the zip file
    private static final int MAX_NUM_POLL_REQUESTS = 50;

    private BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

```

```

public static void main(String[] args) throws Exception {
    FileBasedDeployAndRetrieve sample = new FileBasedDeployAndRetrieve();
    sample.run();
}

public FileBasedDeployAndRetrieve() {
}

private void run() throws Exception {
    this.metadataConnection = MetadataLoginUtil.login();

    // Show the options to retrieve or deploy until user exits
    String choice = getUsersChoice();
    while (choice != null && !choice.equals("99")) {
        if (choice.equals("1")) {
            retrieveZip();
        } else if (choice.equals("2")) {
            deployZip();
        } else {
            break;
        }
        // show the options again
        choice = getUsersChoice();
    }
}

/*
 * Utility method to present options to retrieve or deploy.
 */
private String getUsersChoice() throws IOException {
    System.out.println(" 1: Retrieve");
    System.out.println(" 2: Deploy");
    System.out.println("99: Exit");
    System.out.println();
    System.out.print("Enter 1 to retrieve, 2 to deploy, or 99 to exit: ");
    // wait for the user input.
    String choice = reader.readLine();
    return choice != null ? choice.trim() : "";
}

private void deployZip() throws Exception {
    byte zipBytes[] = readZipFile();
    DeployOptions deployOptions = new DeployOptions();
    deployOptions.setPerformRetrieve(false);
    deployOptions.setRollbackOnError(true);
    AsyncResult asyncResult = metadataConnection.deploy(zipBytes, deployOptions);
    DeployResult result = waitForDeployCompletion(asyncResult.getId());
    if (!result.isSuccess()) {
        printErrors(result, "Final list of failures:\n");
        throw new Exception("The files were not successfully deployed");
    }
    System.out.println("The file " + ZIP_FILE + " was successfully deployed\n");
}

/*
 * Read the zip file contents into a byte array.
 */
private byte[] readZipFile() throws Exception {
    byte[] result = null;
    // We assume here that you have a deploy.zip file.
    // See the retrieve sample for how to retrieve a zip file.
    File zipFile = new File(ZIP_FILE);
    if (!zipFile.exists() || !zipFile.isFile()) {
        throw new Exception("Cannot find the zip file for deploy() on path:"
            + zipFile.getAbsolutePath());
    }

    FileInputStream fileInputStream = new FileInputStream(zipFile);
    try {

```

```

        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        byte[] buffer = new byte[4096];
        int bytesRead = 0;
        while (-1 != (bytesRead = fileInputStream.read(buffer))) {
            bos.write(buffer, 0, bytesRead);
        }

        result = bos.toByteArray();
    } finally {
        fileInputStream.close();
    }
    return result;
}

/*
 * Print out any errors, if any, related to the deploy.
 * @param result - DeployResult
 */
private void printErrors(DeployResult result, String messageHeader) {
    DeployDetails details = result.getDetails();
    StringBuilder stringBuilder = new StringBuilder();
    if (details != null) {
        DeployMessage[] componentFailures = details.getComponentFailures();
        for (DeployMessage failure : componentFailures) {
            String loc = "(" + failure.getLineNumber() + ", " + failure.getColumnNumber();

            if (loc.length() == 0 && !failure.getFileName().equals(failure.getFullName()))
            {
                loc = "(" + failure.getFullName() + ")";
            }
            stringBuilder.append(failure.getFileName() + loc + ":"
                + failure.getProblem()).append('\n');
        }
        RunTestsResult rtr = details.getRunTestResult();
        if (rtr.getFailures() != null) {
            for (RunTestFailure failure : rtr.getFailures()) {
                String n = (failure.getNamespace() == null ? "" :
                    (failure.getNamespace() + ".")) + failure.getName();
                stringBuilder.append("Test failure, method: " + n + "." +
                    failure.getMethodName() + " -- " + failure.getMessage() +
                    " stack " + failure.getStackTrace() + "\n\n");
            }
        }
        if (rtr.getCodeCoverageWarnings() != null) {
            for (CodeCoverageWarning ccw : rtr.getCodeCoverageWarnings()) {
                stringBuilder.append("Code coverage issue");
                if (ccw.getName() != null) {
                    String n = (ccw.getNamespace() == null ? "" :
                        (ccw.getNamespace() + ".")) + ccw.getName();
                    stringBuilder.append(", class: " + n);
                }
                stringBuilder.append(" -- " + ccw.getMessage() + "\n\n");
            }
        }
    }
    if (stringBuilder.length() > 0) {
        stringBuilder.insert(0, messageHeader);
        System.out.println(stringBuilder.toString());
    }
}

private void retrieveZip() throws Exception {
    RetrieveRequest retrieveRequest = new RetrieveRequest();
    retrieveRequest.setApiVersion(API_VERSION);
    setUnpackaged(retrieveRequest);

    AsyncResult asyncResult = metadataConnection.retrieve(retrieveRequest);
    asyncResult = waitForRetrieveCompletion(asyncResult);
    RetrieveResult result =

```



```

        metadataConnection.checkRetrieveStatus(asyncResult.getId());

// Print out any warning messages
StringBuilder stringBuilder = new StringBuilder();
if (result.getMessages() != null) {
    for (RetrieveMessage rm : result.getMessages()) {
        stringBuilder.append(rm.getFileName() + " - " + rm.getProblem() + "\n");
    }
}
if (stringBuilder.length() > 0) {
    System.out.println("Retrieve warnings:\n" + stringBuilder);
}

System.out.println("Writing results to zip file");
File resultsFile = new File(ZIP_FILE);
FileOutputStream os = new FileOutputStream(resultsFile);

try {
    os.write(result.getZipFile());
} finally {
    os.close();
}
}

private DeployResult waitForDeployCompletion(String asyncResultId) throws Exception {
    int poll = 0;
    long waitTimeMilliSecs = ONE_SECOND;
    DeployResult deployResult;
    boolean fetchDetails;
    do {
        Thread.sleep(waitTimeMilliSecs);
        // double the wait time for the next iteration

        waitTimeMilliSecs *= 2;
        if (poll++ > MAX_NUM_POLL_REQUESTS) {
            throw new Exception(
                "Request timed out. If this is a large set of metadata components, " +
                "ensure that MAX_NUM_POLL_REQUESTS is sufficient.");
        }
        // Fetch in-progress details once for every 3 polls
        fetchDetails = (poll % 3 == 0);

        deployResult = metadataConnection.checkDeployStatus(asyncResultId, fetchDetails);

        System.out.println("Status is: " + deployResult.getStatus());
        if (!deployResult.isDone() && fetchDetails) {
            printErrors(deployResult, "Failures for deployment in progress:\n");
        }
    }
    while (!deployResult.isDone());

    if (!deployResult.isSuccess() && deployResult.getErrorStatusCode() != null) {
        throw new Exception(deployResult.getErrorStatusCode() + " msg: " +
            deployResult.getErrorMessage());
    }

    if (!fetchDetails) {
        // Get the final result with details if we didn't do it in the last attempt.
        deployResult = metadataConnection.checkDeployStatus(asyncResultId, true);
    }

    return deployResult;
}

private AsyncResult waitForRetrieveCompletion(AsyncResult asyncResult) throws Exception
{
    int poll = 0;
    long waitTimeMilliSecs = ONE_SECOND;
    while (!asyncResult.isDone()) {
        Thread.sleep(waitTimeMilliSecs);
    }
}

```

```

        // double the wait time for the next iteration

        waitTimeMilliSecs *= 2;
        if (poll++ > MAX_NUM_POLL_REQUESTS) {
            throw new Exception(
                "Request timed out. If this is a large set of metadata components, " +
                "ensure that MAX_NUM_POLL_REQUESTS is sufficient.");
        }

        asyncResult = metadataConnection.checkStatus(
            new String[]{asyncResult.getId()})[0];
        System.out.println("Status is: " + asyncResult.getState());
    }

    if (asyncResult.getState() != AsyncRequestState.Completed) {
        throw new Exception(asyncResult.getStatusCode() + " msg: " +
            asyncResult.getMessage());
    }
    return asyncResult;
}

private void setUnpackaged(RetrieveRequest request) throws Exception {
    // Edit the path, if necessary, if your package.xml file is located elsewhere
    File unpackedManifest = new File(MANIFEST_FILE);
    System.out.println("Manifest file: " + unpackedManifest.getAbsolutePath());

    if (!unpackedManifest.exists() || !unpackedManifest.isFile()) {
        throw new Exception("Should provide a valid retrieve manifest " +
            "for unpackaged content. Looking for " +
            unpackedManifest.getAbsolutePath());
    }

    // Note that we use the fully qualified class name because
    // of a collision with the java.lang.Package class
    com.sforce.soap.metadata.Package p = parsePackageManifest(unpackedManifest);
    request.setUnpackaged(p);
}

private com.sforce.soap.metadata.Package parsePackageManifest(File file)
    throws ParserConfigurationException, IOException, SAXException {
    com.sforce.soap.metadata.Package packageManifest = null;
    List<PackageTypeMembers> listPackageTypes = new ArrayList<PackageTypeMembers>();
    DocumentBuilder db =
        DocumentBuilderFactory.newInstance().newDocumentBuilder();
    InputStream inputStream = new FileInputStream(file);
    Element d = db.parse(inputStream).getDocumentElement();
    for (Node c = d.getFirstChild(); c != null; c = c.getNextSibling()) {
        if (c instanceof Element) {
            Element ce = (Element) c;
            NodeList nodeList = ce.getElementsByTagName("name");
            if (nodeList.getLength() == 0) {
                continue;
            }
            String name = nodeList.item(0).getTextContent();
            NodeList m = ce.getElementsByTagName("members");
            List<String> members = new ArrayList<String>();
            for (int i = 0; i < m.getLength(); i++) {
                Node mm = m.item(i);
                members.add(mm.getTextContent());
            }
            PackageTypeMembers packageTypes = new PackageTypeMembers();
            packageTypes.setName(name);
            packageTypes.setMembers(members.toArray(new String[members.size()]));
            listPackageTypes.add(packageTypes);
        }
    }
    packageManifest = new com.sforce.soap.metadata.Package();
    PackageTypeMembers[] packageTypesArray =
        new PackageTypeMembers[listPackageTypes.size()];
    packageManifest.setTypes(listPackageTypes.toArray(packageTypesArray));
}

```

```
packageManifest.setVersion(API_VERSION + "");  
return packageManifest;  
}  
}
```

USING METADATA API

Chapter 3

Deploying and Retrieving Metadata

Use the `deploy()` and `retrieve()` calls to move metadata (XML files) between a Database.com organization and a local file system. Once you retrieve your XML files into a file system, you can manage changes in a source-code control system, copy and paste code or setup configurations, diff changes to components, and perform many other file-based development operations. At any time you can deploy those changes to another Database.com organization.



Note: The Force.com IDE and the Force.com Migration Tool use the `deploy()` and `retrieve()` calls to move metadata. If you use these tools, interaction with Metadata API is seamless and invisible. Therefore, most developers will find it much easier to use these tools than write code that calls `deploy()` and `retrieve()` directly.

Data in XML files is formatted using the English (United States) locale. This ensures that fields that depend on locale, such as date fields, are interpreted consistently during data migrations between organizations using different languages. Organizations can support multiple languages for presentation to their users.

The `deploy()` and `retrieve()` calls are used primarily for the following development scenarios:

- Development of a custom application (or customization) in a test database organization. After development and testing is completed, the application or customization is then deployed into a production organization using Metadata API.

See Also:

[Metadata Components and Types](#)

Working with the Zip File

The `deploy()` and `retrieve()` calls are used to deploy and retrieve a .zip file. Within the .zip file is a project manifest (`package.xml`) that lists what to retrieve or deploy, and one or more XML components organized into folders.



Note: A component is an instance of a metadata type. For example, `CustomObject` is a metadata type for custom objects, and the `MyCustomObject__c` component is an instance of a custom object.

The files retrieved or deployed in a .zip file may be unpackaged components that reside in your organization (such as *standard objects*), or packaged components that reside within named packages.



Note: Metadata API can deploy and retrieve up to 5,000 files at one time. While a specific file size limit is not enforced, you might encounter out-of-memory errors for very large files.

Every .zip file contains a project manifest, a file named `package.xml`, and a set of directories that contain the components. The manifest file defines the components you are trying to retrieve or deploy in the .zip file.

The following is a sample `package.xml` file. Note that you can retrieve an individual component for a metadata type by specifying its `fullName` field value in a `members` element, or you can also retrieve all components of a metadata type, by using `<members>*</members>`.

```
<?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>
  <types>
    <members>*</members>
    <name>CustomTab</name>
  </types>
  <types>
    <members>Standard</members>
    <name>Profile</name>
  </types>
  <version>30.0</version>
</Package>
```

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

- `<fullName>` contains the name of the server-side package. If no `<fullName>` exists, this is a client-side unpackaged package.
- `<types>` contains the name of the metadata type (for example, `CustomObject`) and the named members (for example, `myCustomObject__c`), to be retrieved or deployed. There can be multiple `<types>` elements in a manifest file and there is one entry for each named component, and one entry for each individual member.
- `<members>` contains the `fullName` of the component, for example `MyCustomObject__c`. The `listMetadata()` call is useful to find out the `fullName` for components of a particular metadata type, if you want to retrieve an individual component. For many metadata types, you can replace the value in `members` with the wildcard character `*` (asterisk) instead of listing each member separately. For a list of metadata types that allow the wildcard character, see the “Allows Wildcard (*)?” column in [Metadata Types](#).



Note: You specify Security in the `<members>` element and Settings in the name element when retrieving the SecuritySettings component type.

- `<name>` contains the metadata type, for example `CustomObject` or `Profile`. There is one name defined for each metadata type in the directory. Any metadata type that extends [Metadata](#) is a valid value. The name entered must match a metadata type defined in the Metadata API WSDL. See [Metadata Types](#) for a list.
- `<version>` is the API version number used when deploying or retrieving the `.zip` file. Currently the valid value is `30.0`.

For more sample `package.xml` manifest files that show you how to work with different subsets of metadata, see [Sample package.xml Manifest Files](#).

To delete items, use the same procedure, but also include a delete manifest file named `destructiveChanges.xml`. To bypass the Recycle Bin, see [purgeOnDelete](#).

The format of `destructiveChanges.xml` is the same as `package.xml`, except that wildcards are not supported.



Note: If you try to delete some components that do not exist in the organization, the rest of the deletions are still 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>30.0</version>
</Package>

```

See Also:

[Metadata Types](#)

Sample package.xml Manifest Files

This section includes sample `package.xml` manifest files that show you how to work with different subsets of metadata. A manifest file can include multiple `<types>` elements so you could combine the individual samples into one `package.xml` manifest file if you want to work with all the metadata in one batch. For more information about the structure of a manifest file, see [Working with the Zip File](#). The following samples are listed:

- [All Custom Objects](#)
- [Custom Fields](#)
- [Packages](#)
- [Security Settings](#)
- [Assignment Rules, Auto-response Rules, Escalation Rules](#)
- [Managed Component Access](#) on page 18

All Custom Objects

This sample `package.xml` manifest file illustrates how to work with 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>30.0</version>
</Package>

```

This manifest file can be used to retrieve or deploy all custom objects. This does not include all standard objects.

Custom Fields

This sample `package.xml` manifest file illustrates how to work with custom fields in custom and standard objects.

```

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

```

```

        <members>Account.SLA__c</members>
        <name>CustomField</name>
    </types>
    <version>30.0</version>
</Package>

```

Note the *objectName.customField* syntax in the `<members>` field where *objectName* is the name of the object, such as `Account`, and *customField* is the name of the custom field, such as an `SLA` picklist field representing a service-level agreement option. The `MyCustomField` custom field in the `MyCustomObject` custom object is uniquely identified by its full name, `MyCustomObject__c.MyCustomField__c`.

Packages

To retrieve a package, set the name of the package in the `packageName` field in `RetrieveRequest` when you call `retrieve()`. The `package.xml` manifest file is automatically populated in the retrieved `.zip` file. The `<fullName>` element in `package.xml` contains the name of the retrieved package.

If you use an asterisk wildcard in a `<members>` element to retrieve all the components of a particular metadata type, the retrieved contents do not include components in managed packages. For more information about managed packages, see the [ISVforce Guide](#).

The easiest way to retrieve a component in a managed package is to retrieve the complete package by setting the name of the package in the `packageName` field in `RetrieveRequest`, as described above. The following sample `package.xml` manifest file illustrates an alternative to retrieve an individual component in a package.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>myns__MyCustomObject__c</members>
        <name>CustomObject</name>
    </types>
    <version>30.0</version>
</Package>

```

Note the *namespacePrefix__objectName* syntax in the `<members>` field where *namespacePrefix* is the namespace prefix of the package and *objectName* is the name of the object. A namespace prefix is a one to 15-character alphanumeric identifier that distinguishes your package and its contents from packages of other publishers. For more information about namespace prefixes, see “Registering a Namespace Prefix” in the Database.com online help.

Security Settings

This sample `package.xml` manifest file illustrates how to work with an organization’s security settings. You specify `Security` in the `<members>` element and `Settings` in the name element when retrieving the `SecuritySettings` component type.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Security</members>
        <name>Settings</name>
    </types>
    <version>30.0</version>
</Package>

```

Assignment Rules, Auto-response Rules, Escalation Rules

Assignment rules, auto-response rules and escalation rules use different `package.xml` type names to access sets of rules or individual rules for object types. For example, the following sample `package.xml` manifest file illustrates how to access an organization’s assignment rules for just `Cases` and `Leads`.

```

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

```

```

    <types>
      <members>Case</members>
      <members>Lead</members>
      <name>AssignmentRules</name>
    </types>
    <version>30.0</version>
  </Package>

```

The following sample package.xml manifest file illustrates how to access just the “samplerule” Case assignment rule and the “newrule” Lead assignment rule. Notice that the type name is AssignmentRule and not AssignmentRules.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>Case.samplerule</members>
    <members>Lead.newrule</members>
    <name>AssignmentRule</name>
  </types>
  <version>30.0</version>
</Package>

```

Similarly, for accessing individual auto-response rules and escalation rules, use AutoResponseRule and EscalationRule instead of AutoResponseRules and EscalationRules.

Managed Component Access

In API version 29.0 and later, you can retrieve and deploy access settings for the following managed components in profiles and permission sets:

- Apex classes
- Custom field permissions
- Custom object permissions
- External data sources

When retrieving and deploying managed component permissions, specify the namespace followed by two underscores. Wildcards are not supported.

For example, let’s say you install a managed package with the namespace MyNamespace and the custom object JobRequest__c. To set object permissions for JobRequest__c in the package to the custom profile MyProfile, you would add the following to the .profile file.

To deploy:

```

<objectPermissions>
  <allowCreate>true</allowCreate>
  <allowDelete>true</allowDelete>
  <allowEdit>true</allowEdit>
  <allowRead>true</allowRead>
  <viewAllRecords>>false</viewAllRecords>
  <modifyAllRecords>>false</modifyAllRecords>
  <object>MyNamespace__JobRequest__c</object>
</objectPermissions>

```

To retrieve:

```

<types>
  <members>MyNamespace__JobRequest__c</members>
  <name>CustomObject</name>
</types>
<types>
  <members>MyProfile</members>
  <name>Profile</name>
</types>

```


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.

If the deployment includes components for any of the following metadata types, all the tests are automatically run.

- ApexClass
- ApexComponent
- ApexPage
- ApexTrigger
- ArticleType
- BaseSharingRule
- CriteriaBasedSharingRule
- CustomDataType
- CustomField
- CustomObject
- DataCategoryGroup
- Flow
- InstalledPackage
- NamedFilter
- OwnerSharingRule
- PermissionSet
- Profile
- Queue
- RecordType
- RemoteSiteSetting
- Role
- SharingReason
- Territory
- Validation Rules
- Workflow

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

- One CustomApplication 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

See Also:

[deploy\(\)](#)

Maintaining User References

User fields are preserved during a metadata deployment.

When a component in your deployment refers to a specific user, such as a recipient of a workflow email notification or a dashboard running user, then Database.com attempts to locate a matching user in the destination organization by comparing usernames during the deployment.

For example, when you copy data to a test database, the fields containing usernames from the production organization are altered to include the test database name. In a test database named `test`, the username `user@acme.com` becomes `user@acme.com.test`. When you deploy the metadata in the test database to another organization, the `test` in the username is ignored.

For user references in deployments, Database.com performs the following sequence:

1. Database.com compares usernames in the source environment to the destination environment and adapts the organization domain name.
2. If two or more usernames match, Database.com lists the matching names and requests one of the users in the source environment be renamed.
3. If a username in the source environment doesn't exist in the destination environment, Database.com displays an error, and the deployment stops until the usernames are removed or resolved to users in the destination environment.

Chapter 4

CRUD-Based Metadata Development

Use the CRUD-based metadata calls to create, update, or delete setup and configuration components for your organization or application. These configuration components include custom objects, custom fields, and other configuration metadata. The metadata calls mimic the behavior in the Database.com user interface for creating, updating, or deleting components. Whatever rules apply there also apply to these calls.

Metadata calls are different from the core, synchronous API calls in the following ways:

- Metadata API calls are available in a separate WSDL. To download the WSDL, log into Database.com, from Setup, click **Develop > API** and click the **Download Metadata WSDL** link.
- After logging in, you must send Metadata API calls to the Metadata API endpoint, which has a different URL than the SOAP API. Retrieve the `metadataServerUrl` from the `LoginResult` returned by your SOAP API `login()` call. For more information about the SOAP API, see the *SOAP API Developer's Guide*.
- Metadata calls are either asynchronous or synchronous. For the asynchronous calls, the results are not returned in a single call. The API core calls are synchronous; the results are returned in one call.
- There are three asynchronous metadata calls with the same name as the corresponding core SOAP API synchronous calls, but with different signatures: `create()`, `update()`, and `delete()`. There is also a special utility call, `checkStatus()`, which you use to poll for the completion of the asynchronous call.

The following development workflow is common for asynchronous CRUD-based metadata calls:

1. The *logged-in user* issues a metadata call, specifying all required fields to be created or updated.
2. Database.com returns an `AsyncResult` object, which is updated with status information for each component as the operation moves from a queue to completed or error state.
3. The logged-in user checks the status values in `AsyncResult` to determine when all the create or update operations are completed.



Note: Metadata API also supports `retrieve()` and `deploy()` calls for retrieving and deploying metadata components. For more information, see [Deploying and Retrieving Metadata](#).

Java Sample Code for CRUD-Based Development

This section walks through a sample Java client application that uses CRUD—based calls. This sample application performs the following main tasks:

1. Uses the `MetadataLoginUtil.java` class to create a Metadata connection. For more information, see [Step 3: Walk Through the Java Sample Code](#).
2. Calls `create()` to create a new custom object.

Database.com returns an `AsyncResult` object for each component you tried to create. The `AsyncResult` object is updated with status information as the operation moves from a queue to completed or error state.

3. Calls `checkStatus()` in a loop until the status value in `AsyncResult` indicates that the create operation is completed.

Note the error handling code that follows each API call.

```

import com.sforce.soap.metadata.*;

/**
 * Sample that logs in and creates a custom object through the metadata api
 */
public class CRUDSample {
    private MetadataConnection metadataConnection;

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;

    public CRUDSample() {
    }

    public static void main(String[] args) throws Exception {
        CRUDSample crudSample = new CRUDSample();
        crudSample.runCreate();
    }

    /**
     * Create a custom object. This method demonstrates usage of the
     * create() and checkStatus() calls.
     *
     * @param uniqueName Custom object name should be unique.
     */
    private void createCustomObject(final String uniqueName) throws Exception {
        final String label = "My Custom Object";
        CustomObject customObject = new CustomObject();
        customObject.setFullName(uniqueName);
        customObject.setDeploymentStatus(DeploymentStatus.Deployed);
        customObject.setDescription("Created by the Metadata API Sample");
        customObject.setLabel(label);
        customObject.setPluralLabel(label + "s");
        customObject.setSharingModel(SharingModel.ReadWrite);

        // The name field appears in page layouts, related lists, and elsewhere.
        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setDescription("The custom object identifier on page layouts, related lists
etc");
        nf.setLabel(label);
        nf.setFullName(uniqueName);
        customObject.setNameField(nf);

        AsyncResult[] asyncResults = metadataConnection.create(
            new CustomObject[]{customObject});
        if (asyncResults == null) {
            System.out.println("The object was not created successfully");
            return;
        }

        long waitTimeMilliSecs = ONE_SECOND;

        // After the create() call completes, we must poll the results of the checkStatus()

        // call until it indicates that the create operation has completed.
        do {
            printAsyncResultStatus(asyncResults);
            waitTimeMilliSecs *= 2;
            Thread.sleep(waitTimeMilliSecs);
            asyncResults = metadataConnection.checkStatus(new
String[]{asyncResults[0].getId()});
        } while (!asyncResults[0].isDone());

        printAsyncResultStatus(asyncResults);
    }
}

```

```
private void printAsyncResultStatus(AsyncResult[] asyncResults) throws Exception {
    if (asyncResults == null || asyncResults.length == 0 || asyncResults[0] == null) {
        throw new Exception("The object status cannot be retrieved");
    }

    AsyncResult asyncResult = asyncResults[0]; //we are creating only 1 metadata object

    if (asyncResult.getStatusCode() != null) {
        System.out.println("Error status code: " +
            asyncResult.getStatusCode());
        System.out.println("Error message: " + asyncResult.getMessage());
    }

    System.out.println("Object with id:" + asyncResult.getId() + " is " +
        asyncResult.getState());
}

private void runCreate() throws Exception {
    metadataConnection = MetadataLoginUtil.login();
    // Custom objects and fields must have __c suffix in the full name.
    final String uniqueObjectName = "MyCustomObject__c";
    createCustomObject(uniqueObjectName);
}
}
```

Chapter 5

Error Handling

Metadata API calls return error information that your client application can use to identify and resolve runtime errors. The Metadata API provides the following types of error handling:

- Since the Metadata API uses the enterprise or partner WSDLs to authenticate, it uses SOAP fault messages defined in those WSDLs for errors resulting from badly formed messages, failed authentication, or similar problems. Each SOAP fault has an associated `ExceptionCode`. For more details, see “Error Handling” in the *SOAP API Developer's Guide*.
- For errors with the asynchronous `create()`, `update()`, and `delete()` calls, see the error status code in the `statusCode` field in the `AsyncResult` object for the associated component.
- For errors with the synchronous CRUD calls, see the error status code in the `statusCode` field of the `Error` object corresponding to each error in the array returned by the `errors` field of the appropriate result object. For example, the result object of `createMetadata()` is `SaveResult`.
- For errors with `deploy()`, see the `problem` and `success` fields in the `DeployMessage` object for the associated component.
- For errors with `retrieve()`, see the `problem` field in the `RetrieveMessage` object for the associated component.

For sample code, see [Step 3: Walk Through the Java Sample Code](#) on page 7.

Error Handling for Session Expiration

When you sign on via the `login()` call, a new client session begins and a corresponding unique session ID is generated. Sessions automatically expire after the amount of time specified in the **Security Controls** setup area of the Database.com application (default two hours). When your session expires, the exception code `INVALID_SESSION_ID` is returned. If this happens, you must invoke the `login()` call again. For more information about `login()`, see the *SOAP API Developer's Guide*.

Use the following [file-based](#) calls to deploy or retrieve XML components.

- `deploy()`
- `retrieve()`

`deploy()`

Uses file representations of components to create, update, or delete those components in an organization.

Syntax

```
AsyncResult = metadatabinding.deploy(base64 zipFile, DeployOptions deployOptions)
```

Usage

Use this call to take file representations of components and deploy them into an organization by creating, updating, or deleting the components they represent.



Note: Metadata API can deploy and retrieve up to 5,000 files at one time. While a specific file size limit is not enforced, you might encounter out-of-memory errors for very large files.

In API version 29.0, Database.com improved the deployment status properties and removed the requirement to use `checkStatus()` after a `deploy()` call to get information about deployments. Database.com continues to support the use of `checkStatus()` when using `deploy()` with API version 28.0 or earlier.

For API version 29.0 or later, `deploy` (create or update) packaged or unpackaged components using the following steps.

1. Issue a `deploy()` call to start the asynchronous deployment. An `AsyncResult` object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkDeployStatus()` call in a loop until the `done` field of the returned `DeployResult` contains `true`, which means that the call is completed. The `DeployResult` object contains information about an in-progress or completed deployment started using the `deploy()` call. When calling `checkDeployStatus()`, pass in the `id` value from the `AsyncResult` object from the first step.

For API version 28.0 or earlier, `deploy` (create or update) packaged or unpackaged components using the following steps.

1. Issue a `deploy()` call to start the asynchronous deployment. An `AsyncResult` object is returned. If the call is completed, the `done` field contains `true`. Most often, the call is not completed quickly enough to be noted in the first result. If it is completed, note the value in the `id` field returned and skip the next step.

- If the call is not complete, issue a `checkStatus()` call in a loop using the value in the `id` field of the `AsyncResult` object returned by the `deploy()` call in the previous step. Check the `AsyncResult` object which is returned until the `done` field contains `true`. The time taken to complete a `deploy()` call depends on the size of the zip file being deployed, so a longer wait time between iterations should be used as the size of the zip file increases.
- Issue a `checkDeployStatus()` call to obtain the results of the `deploy()` call, using the `id` value returned in the first step.

To delete items, use the same procedure, but also include a delete manifest file named `destructiveChanges.xml`. To bypass the Recycle Bin, see [purgeOnDelete](#).

The format of `destructiveChanges.xml` is the same as `package.xml`, except that wildcards are not supported.



Note: If you try to delete some components that do not exist in the organization, the rest of the deletions are still 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>30.0</version>
</Package>
```

To track the status of deployments that are in progress or completed in the last 30 days, from Setup, click **Deployment Status** or **Deploy > Deployment Status**.

You can cancel a deployment while it's in progress or in the queue by clicking **Cancel** next to the deployment. The deployment then has the status `Cancel Requested` until the deployment is completely canceled. A canceled deployment is listed in the `Failed` section.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Arguments

Name	Type	Description
<code>zipFile</code>	<code>base64</code>	Base 64-encoded binary data. Client applications must encode the binary data as base64.
<code>deployOptions</code>	DeployOptions	Encapsulates options for determining which packages or files are deployed.

DeployOptions

The following deployment options can be selected for this call:

Name	Type	Description
<code>allowMissingFiles</code>	boolean	<p>Specifies whether a deploy succeeds even if files that are specified in <code>package.xml</code> but are not in the <code>.zip</code> file (<code>true</code> or not <code>false</code>).</p> <p>Do not set this argument for deployment to <i>production organizations</i>.</p>
<code>autoUpdatePackage</code>	boolean	<p>If a file is in the <code>.zip</code> file but not specified in <code>package.xml</code>, specifies whether the file should be automatically added to the package (<code>true</code> or not <code>false</code>). A <code>retrieve()</code> is automatically issued with the updated <code>package.xml</code> that includes the <code>.zip</code> file.</p> <p>Do not set this argument for deployment to <i>production organizations</i>.</p>
<code>checkOnly</code>	boolean	<p>Indicates whether Apex classes and triggers are saved to the organization as part of the deployment (<code>false</code>) or not (<code>true</code>). Defaults to <code>false</code>. Any errors or messages that would have been issued are still generated. This parameter is similar to the Database.com Ant tool's <code>checkOnly</code> parameter.</p>
<code>ignoreWarnings</code>	boolean	<p>Indicates whether a warning should allow a deployment to complete successfully (<code>true</code>) or not (<code>false</code>). Defaults to <code>false</code>.</p> <p>The <code>DeployMessage</code> object for a warning contains the following values:</p> <ul style="list-style-type: none"> <code>problemType</code>—Warning <code>problem</code>—The text of the warning. <p>If a warning occurs and <code>ignoreWarnings</code> is set to <code>true</code>, the <code>success</code> field in <code>DeployMessage</code> is <code>true</code>. If <code>ignoreWarnings</code> is set to <code>false</code>, <code>success</code> is set to <code>false</code> and the warning is treated like an error.</p> <p>This field is available in API version 18.0 and later. Prior to version 18.0, there was no distinction between warnings and errors. All problems were treated as errors and prevented a successful deployment.</p>
<code>performRetrieve</code>	boolean	<p>Indicates whether a <code>retrieve()</code> call is performed immediately after the deployment (<code>true</code>) or not (<code>false</code>). Set to <code>true</code> in order to retrieve whatever was just deployed.</p>
<code>purgeOnDelete</code>	boolean	<p>If <code>true</code>, the deleted components in the <code>destructiveChanges.xml</code> manifest file aren't stored in the Recycle Bin. Instead, they become immediately eligible for deletion.</p> <p>This field is available in API version 22.0 and later.</p> <p>This option only works in test database organizations; it doesn't work in production organizations.</p>
<code>rollbackOnError</code>	boolean	<p>Indicates whether any failure causes a complete rollback (<code>true</code>) or not (<code>false</code>). If <code>false</code>, whatever set of actions</p>

Name	Type	Description
		can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to <code>true</code> if you are deploying to a production organization. The default is <code>false</code> .
<code>runAllTests</code>	<code>boolean</code>	If <code>true</code> , all Apex tests defined in the organization are run. 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 the tests fail when the <code>rollbackOnError</code> parameter is set to <code>true</code> , the entire deployment will roll back.
<code>runTests</code>	<code>string[]</code>	A list of Apex tests to be run during deployment. Specify the class name, one name per instance. The class name may also specify a namespace with a dot. For example, to run three tests: <pre><runTests>positive_test</runTests> <runTests>negative_test</runTests> <runTests>namespace.third_test</runTests></pre> If any of these tests fail when the <code>rollbackOnError</code> parameter is set to <code>true</code> , the deployment is rolled back and no changes will be made to your organization.
<code>singlePackage</code>	<code>boolean</code>	Indicates whether the specified <code>.zip</code> file points to a directory structure with a single package (<code>true</code>) or a set of packages (<code>false</code>).

Response

[AsyncResult](#)

Sample Code—Java

This sample shows how to deploy components in a zip file. See the [retrieve\(\) sample code](#) for details on how to retrieve a zip file.

```
package com.doc.samples;

import java.io.*;

import java.rmi.RemoteException;

import com.sforce.soap.metadata.AsyncResult;
import com.sforce.soap.metadata.DeployDetails;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.soap.metadata.DeployOptions;
import com.sforce.soap.metadata.DeployResult;
import com.sforce.soap.metadata.DeployMessage;
import com.sforce.soap.metadata.RunTestsResult;
import com.sforce.soap.metadata.RunTestFailure;
import com.sforce.soap.metadata.CodeCoverageWarning;
import com.sforce.soap.partner.LoginResult;
import com.sforce.soap.partner.PartnerConnection;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;
```

```

/**
 * Deploy a zip file of metadata components.
 * Prerequisite: Have a deploy.zip file that includes a package.xml manifest file that
 * details the contents of the zip file.
 */
public class DeploySample {
    // binding for the metadata WSDL used for making metadata API calls
    private MetadataConnection metadataConnection;

    static BufferedReader rdr = new BufferedReader(new InputStreamReader(System.in));

    private static final String ZIP_FILE = "deploy.zip";

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;
    // maximum number of attempts to deploy the zip file
    private static final int MAX_NUM_POLL_REQUESTS = 50;

    public static void main(String[] args) throws Exception {
        final String USERNAME = "user@company.com";
        // This is only a sample. Hard coding passwords in source files is a bad practice.
        final String PASSWORD = "password";
        final String URL = "https://login.salesforce.com/services/Soap/u/29.0";

        DeploySample sample = new DeploySample(USERNAME, PASSWORD, URL);
        sample.deployZip();
    }

    public DeploySample(String username, String password, String loginUrl) throws
    ConnectionException {
        createMetadataConnection(username, password, loginUrl);
    }

    public void deployZip()
    throws RemoteException, Exception
    {
        byte zipBytes[] = readZipFile();
        DeployOptions deployOptions = new DeployOptions();
        deployOptions.setPerformRetrieve(false);
        deployOptions.setRollbackOnError(true);
        AsyncResult asyncResult = metadataConnection.deploy(zipBytes, deployOptions);
        String asyncResultId = asyncResult.getId();

        // Wait for the deploy to complete
        int poll = 0;
        long waitTimeMilliSecs = ONE_SECOND;
        DeployResult deployResult = null;
        boolean fetchDetails;
        do {
            Thread.sleep(waitTimeMilliSecs);
            // double the wait time for the next iteration
            waitTimeMilliSecs *= 2;
            if (poll++ > MAX_NUM_POLL_REQUESTS) {
                throw new Exception("Request timed out. If this is a large set " +
                    "of metadata components, check that the time allowed by " +
                    "MAX_NUM_POLL_REQUESTS is sufficient.");
            }

            // Fetch in-progress details once for every 3 polls
            fetchDetails = (poll % 3 == 0);
            deployResult = metadataConnection.checkDeployStatus(asyncResultId, fetchDetails);

            System.out.println("Status is: " + deployResult.getStatus());
            if (!deployResult.isDone() && fetchDetails) {
                printErrors(deployResult, "Failures for deployment in progress:\n");
            }
        }
        while (!deployResult.isDone());
    }
}

```

```

    if (!deployResult.isSuccess() && deployResult.getErrorStatusCode() != null) {
        throw new Exception(deployResult.getErrorStatusCode() + " msg: " +
            deployResult.getErrorMessage());
    }

    if (!fetchDetails) {
        // Get the final result with details if we didn't do it in the last attempt.
        deployResult = metadataConnection.checkDeployStatus(asyncResultId, true);
    }

    if (!deployResult.isSuccess()) {
        printErrors(deployResult, "Final list of failures:\n");
        throw new Exception("The files were not successfully deployed");
    }

    System.out.println("The file " + ZIP_FILE + " was successfully deployed");
}

/**
 * Read the zip file contents into a byte array.
 * @return byte[]
 * @throws Exception - if cannot find the zip file to deploy
 */
private byte[] readZipFile()
    throws Exception
{
    // We assume here that you have a deploy.zip file.
    // See the retrieve sample for how to retrieve a zip file.
    File deployZip = new File(ZIP_FILE);
    if (!deployZip.exists() || !deployZip.isFile())
        throw new Exception("Cannot find the zip file to deploy. Looking for " +
            deployZip.getAbsolutePath());

    FileInputStream fos = new FileInputStream(deployZip);
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    int readbyte = -1;
    while ((readbyte = fos.read()) != -1) {
        bos.write(readbyte);
    }
    fos.close();
    bos.close();
    return bos.toByteArray();
}

/**
 * Print out any errors, if any, related to the deploy.
 * @param result - DeployResult
 */
private void printErrors(DeployResult result, String messageHeader)
{
    DeployDetails deployDetails = result.getDetails();

    StringBuilder errorMessageBuilder = new StringBuilder();
    if (deployDetails != null) {
        DeployMessage[] componentFailures = deployDetails.getComponentFailures();
        for (DeployMessage message : componentFailures) {
            String loc = (message.getLineNumber() == 0 ? "" :
                "(" + message.getLineNumber() + ", " +
                    message.getColumnNumber() + ")");
            if (loc.length() == 0
                && !message.getFileName().equals(message.getFullName())) {
                loc = "(" + message.getFullName() + ")";
            }
            errorMessageBuilder.append(message.getFileName() + loc + ":" +
                message.getProblem()).append('\n');
        }
        RunTestsResult rtr = deployDetails.getRunTestResult();
        if (rtr.getFailures() != null) {
            for (RunTestFailure failure : rtr.getFailures()) {

```

```

        String n = (failure.getNamespace() == null ? "" :
            (failure.getNamespace() + ".")) + failure.getName();
        errorMessageBuilder.append("Test failure, method: " + n + "." +
            failure.getMethodName() + " -- " +
            failure.getMessage() + " stack " +
            failure.getStackTrace() + "\n\n");
    }
}
if (rtr.getCodeCoverageWarnings() != null) {
    for (CodeCoverageWarning ccw : rtr.getCodeCoverageWarnings()) {
        errorMessageBuilder.append("Code coverage issue");
        if (ccw.getName() != null) {
            String n = (ccw.getNamespace() == null ? "" :
                (ccw.getNamespace() + ".")) + ccw.getName();
            errorMessageBuilder.append(", class: " + n);
        }
        errorMessageBuilder.append(" -- " + ccw.getMessage() + "\n");
    }
}
}

if (errorMessageBuilder.length() > 0) {
    errorMessageBuilder.insert(0, messageHeader);
    System.out.println(errorMessageBuilder.toString());
}
}

private void createMetadataConnection(
    final String username,
    final String password,
    final String loginUrl) throws ConnectionException {

    final ConnectorConfig loginConfig = new ConnectorConfig();
    loginConfig.setAuthEndpoint(loginUrl);
    loginConfig.setServiceEndpoint(loginUrl);
    loginConfig.setManualLogin(true);
    LoginResult loginResult = (new PartnerConnection(loginConfig)).login(username,
password);

    final ConnectorConfig metadataConfig = new ConnectorConfig();
    metadataConfig.setServiceEndpoint(loginResult.getMetadataServerUrl());
    metadataConfig.setSessionId(loginResult.getSessionId());
    this.metadataConnection = new MetadataConnection(metadataConfig);
}
}
}

```

See Also:[Running Tests in a Deployment](#)**checkDeployStatus ()**

Checks the status of declarative metadata call `deploy()`.

Syntax

```
DeployResult = metadataBinding.checkDeployStatus(ID id, includeDetails boolean);
```

Usage

`checkDeployStatus` is used as part of the process for deploying packaged or unpackaged components to an organization:

1. Issue a `deploy()` call to start the asynchronous deployment. An `AsyncResult` object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkDeployStatus()` call in a loop until the `done` field of the returned `DeployResult` contains `true`, which means that the call is completed. The `DeployResult` object contains information about an in-progress or completed deployment started using the `deploy()` call. When calling `checkDeployStatus()`, pass in the `id` value from the `AsyncResult` object from the first step.

In API version 29.0, Database.com improved the deployment status properties and removed the requirement to use `checkStatus()` after a `deploy()` call to get information about deployments. Database.com continues to support the use of `checkStatus()` when using `deploy()` with API version 28.0 or earlier.

Sample Code—Java

See the [deploy\(\) sample code](#) for sample usage of this call.

Arguments

Name	Type	Description
<code>id</code>	ID	ID obtained from an <code>AsyncResult</code> object returned by <code>deploy()</code> or a subsequent <code>checkDeployStatus()</code> call.
<code>includeDetails</code>	boolean	Sets the <code>DeployResult</code> object to include <code>DeployDetails</code> information (<code>true</code>) or not (<code>false</code>). The default is <code>false</code> . Available in API version 29.0 and later.

Response

`DeployResult`

cancelDeploy()

Cancel a deployment that hasn't completed yet.

Syntax

```
CancelDeployResult = metadatabinding.cancelDeploy(string id)
```

Usage

Use the `cancelDeploy()` operation to cancel a deployment in your organization started by the `deploy()` operation, which includes deployments started by the Force.com Migration Tool and the Force.com IDE. The deployment can be in a queue waiting to get started, or can be in progress. This operation takes the ID of the deployment you wish to cancel and returns a `CancelDeployResult` object. When the deployment is in the queue and hasn't started yet, calling `cancelDeploy()` cancels the deployment immediately. When the deployment has started and is in progress, it might not get canceled immediately, so you should call `checkDeployStatus()` to check the status of the cancellation.

Cancel a deployment using these steps.

1. Obtain the ID of the deployment you wish to cancel. For example, you can obtain the ID from the `deploy()` call in the `AsyncResult` object `id` field. Alternatively, you can obtain the ID in the Database.com user interface from Setup by clicking **Deployment Status** or **Deploy > Deployment Status**, and by noting the ID of a deployment started by the API.
2. Issue a `cancelDeploy()` call to start the cancellation process. This call returns a `CancelDeployResult` object.
3. Check the value in the `done` field of the returned `CancelDeployResult`. If the `done` field value is `true`, the deployment has been canceled and you're done. If the `done` field value is `false`, the cancellation is in progress, and follow these steps to check the cancellation status.

- a. Call `checkDeployStatus()` using the deployment ID you obtained earlier.
- b. In the returned `DeployResult` object, check the `status` field. If the status is `Canceling`, this means the cancellation is still in progress, and repeat steps a and b. Otherwise, if the status is `Canceled`, this means the deployment has been canceled and you're done.

The `deploy()` operation throws these API faults.

INVALID_ID_FIELD with the message `Invalid deploy ID`

The specified ID argument doesn't correspond to a valid deployment.

INVALID_ID_FIELD with the message `Deployment already completed`

The specified deployment has already completed.

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the "Modify All Data" permission.

Arguments

Name	Type	Description
<code>id</code>	string	The ID of the deployment to cancel.

Response

`CancelDeployResult`

Sample Code—Java

This sample shows how to cancel a deployment. The sample calls `cancelDeploy()` by passing it a given deployment ID. Next, it checks whether the cancellation has completed, and if not, calls `checkDeployStatus` in a loop.

```
public void cancelDeploy(String asyncId) throws Exception {
    // Issue the deployment cancellation request
    CancelDeployResult result = metadataConnection.cancelDeploy(asyncId);

    // If the deployment cancellation completed, write a message to the output.
    if (result.isDone()) {
        System.out.println("Your deployment was canceled successfully!");
    }
    else {
        // The deployment cancellation is still in progress, so get a new status
        DeployResult deployResult = metadataConnection.checkDeployStatus(asyncId, false);

        // Check whether the deployment is done. If not done, this means
        // that the cancellation is still in progress and the status is Canceling.
        while (!deployResult.isDone()) {
            // Assert that the deployment status is Canceling
            assert deployResult.getStatus() == DeployStatus.Canceling;
            // Wait 2 seconds
            Thread.sleep(2000);
            // Get the deployment status again
            deployResult = metadataConnection.checkDeployStatus(asyncId, false);
        }

        // The deployment is done. Write the status to the output.
        // (When the deployment is done, the cancellation should have completed

```

```

    // and the status should be Canceled. However, in very rare cases,
    // the deployment can complete before it is canceled.)
    System.out.println("Final deploy status = >" + deployResult.getStatus());
}
}

```

retrieve()

This call retrieves XML file representations of components in an organization.

Syntax

```
AsyncResult = metadatabinding.retrieve(RetrieveRequest retrieveRequest)
```

Usage

Use this call to retrieve file representations of components in an organization.



Note: Metadata API can deploy and retrieve up to 5,000 files at one time. While a specific file size limit is not enforced, you might encounter out-of-memory errors for very large files.

To retrieve packaged or unpackaged components:

1. Issue a `retrieve()` call to start the asynchronous retrieval. An `AsyncResult` object is returned. If the call is completed, the `done` field contains `true`. Most often, the call is not completed quickly enough to be noted in the result. If it is completed, note the value in the `id` field returned and skip the next step.
2. If the call is not complete, issue a `checkStatus()` call in a loop using the value in the `id` field of the `AsyncResult` object, returned by the `retrieve()` call in the previous step. Check the `AsyncResult` object returned until the `done` field contains `true`. The time taken to complete a `retrieve()` call depends on the size of the zip file being deployed, so use a longer wait time between iterations as the size of the zip file increases.
3. Issue a `checkRetrieveStatus()` call to obtain the results of the `retrieve()` call, using the `id` value returned in the first step.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Arguments

Name	Type	Description
<code>retrieveRequest</code>	<code>RetrieveRequest</code>	Encapsulates options for determining which packages or files are retrieved.

Response

`AsyncResult`

Sample Code—Java

This sample shows how to retrieve components into a zip file. See the `deploy()` [sample code](#) for details on how to deploy a zip file.



Note: This sample was created using Apache Axis. The WSDL2Java utility generates a `_package` class, even though the metadata type is defined as `Package` in the Metadata WSDL. Other SOAP clients might generate a different name for the `_package` class.

```

package com.doc.samples;

import java.io.*;
import java.util.*;
import java.nio.ByteBuffer;
import java.nio.channels.Channels;
import java.nio.channels.FileChannel;
import java.nio.channels.ReadableByteChannel;
import java.nio.channels.WritableByteChannel;

import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.sforce.soap.enterprise.LoginResult;
import com.sforce.soap.enterprise.SessionHeader;
import com.sforce.soap.enterprise.SforceServiceLocator;
import com.sforce.soap.enterprise.SoapBindingStub;
import com.sforce.soap.enterprise.fault.ExceptionCode;
import com.sforce.soap.enterprise.fault.LoginFault;

import com.sforce.soap._2006._04.metadata.MetadataBindingStub;
import com.sforce.soap._2006._04.metadata.MetadataServiceLocator;
import com.sforce.soap._2006._04.metadata.AsyncResult;
import com.sforce.soap._2006._04.metadata.RetrieveRequest;
import com.sforce.soap._2006._04.metadata.AsyncRequestState;
import com.sforce.soap._2006._04.metadata.RetrieveResult;
import com.sforce.soap._2006._04.metadata.RetrieveMessage;
// Note that Axis generates a _package class, even though it is defined as Package
// in the WSDL. Other SOAP clients may generate a different name for the _package class.
import com.sforce.soap._2006._04.metadata._package;
import com.sforce.soap._2006._04.metadata.PackageTypeMembers;

public class RetrieveSample {
    // binding for the Enterprise WSDL used for login() call
    private SoapBindingStub binding;
    // binding for the metadata WSDL used for create() and checkStatus() calls
    private MetadataBindingStub metadatabinding;

    static BufferedReader rdr = new BufferedReader(new InputStreamReader(System.in));

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;
    // maximum number of attempts to retrieve the results
    private static final int MAX_NUM_POLL_REQUESTS = 50;

    // manifest file that controls which components get retrieved
    private static final String MANIFEST_FILE = "package.xml";

    private static final double API_VERSION = 15.0;

    public static void main(String[] args) throws ServiceException, Exception {
        RetrieveSample sample = new RetrieveSample();
        sample.run();
    }
}

```

```

private void run() throws ServiceException, Exception {
    if (login()) {
        getUserInput("SUCCESSFUL LOGIN! Hit the enter key to continue.");
        retrieveZip();
    }
}

private void retrieveZip() throws RemoteException, Exception
{
    RetrieveRequest retrieveRequest = new RetrieveRequest();
    retrieveRequest.setApiVersion(API_VERSION);
    setUnpackaged(retrieveRequest);

    AsyncResult asyncResult = metadatabinding.retrieve(retrieveRequest);
    // Wait for the retrieve to complete
    int poll = 0;
    long waitTimeMilliSecs = ONE_SECOND;
    while (!asyncResult.isDone()) {
        Thread.sleep(waitTimeMilliSecs);
        // double the wait time for the next iteration
        waitTimeMilliSecs *= 2;
        if (poll++ > MAX_NUM_POLL_REQUESTS) {
            throw new Exception("Request timed out. If this is a large set " +
                "of metadata components, check that the time allowed " +
                "by MAX_NUM_POLL_REQUESTS is sufficient.");
        }
        asyncResult = metadatabinding.checkStatus(
            new String[] {asyncResult.getId()})[0];
        System.out.println("Status is: " + asyncResult.getState());
    }

    if (asyncResult.getState() != AsyncRequestState.Completed) {
        throw new Exception(asyncResult.getStatusCode() + " msg: " +
            asyncResult.getMessage());
    }

    RetrieveResult result = metadatabinding.checkRetrieveStatus(asyncResult.getId());

    // Print out any warning messages
    StringBuilder buf = new StringBuilder();
    if (result.getMessages() != null) {
        for (RetrieveMessage rm : result.getMessages()) {
            buf.append(rm.getFileName() + " - " + rm.getProblem());
        }
    }
    if (buf.length() > 0) {
        System.out.println("Retrieve warnings:\n" + buf);
    }

    // Write the zip to the file system
    System.out.println("Writing results to zip file");
    ByteArrayInputStream bais = new ByteArrayInputStream(result.getZipFile());
    File resultsFile = new File("retrieveResults.zip");
    FileOutputStream os = new FileOutputStream(resultsFile);
    try {
        ReadableByteChannel src = Channels.newChannel(bais);
        FileChannel dest = os.getChannel();
        copy(src, dest);

        System.out.println("Results written to " + resultsFile.getAbsolutePath());
    }
    finally {
        os.close();
    }
}

/**

```

```

* Helper method to copy from a readable channel to a writable channel,
* using an in-memory buffer.
*/
private void copy(ReadableByteChannel src, WritableByteChannel dest)
    throws IOException
{
    // use an in-memory byte buffer
    ByteBuffer buffer = ByteBuffer.allocate(8092);
    while (src.read(buffer) != -1) {
        buffer.flip();
        while (buffer.hasRemaining()) {
            dest.write(buffer);
        }
        buffer.clear();
    }
}

private void setUnpackaged(RetrieveRequest request) throws Exception
{
    // Edit the path, if necessary, if your package.xml file is located elsewhere
    File unpackedManifest = new File(MANIFEST_FILE);
    System.out.println("Manifest file: " + unpackedManifest.getAbsolutePath());

    if (!unpackedManifest.exists() || !unpackedManifest.isFile())
        throw new Exception("Should provide a valid retrieve manifest " +
            "for unpackaged content. " +
            "Looking for " + unpackedManifest.getAbsolutePath());

    // Note that we populate the _package object by parsing a manifest file here.
    // You could populate the _package based on any source for your
    // particular application.
    _package p = parsePackage(unpackedManifest);
    request.setUnpackaged(p);
}

private _package parsePackage(File file) throws Exception {
    try {
        InputStream is = new FileInputStream(file);
        List<PackageTypeMembers> pd = new ArrayList<PackageTypeMembers>();
        DocumentBuilder db =
            DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Element d = db.parse(is).getDocumentElement();
        for (Node c = d.getFirstChild(); c != null; c = c.getNextSibling()) {
            if (c instanceof Element) {
                Element ce = (Element)c;
                //
                NodeList namee = ce.getElementsByTagName("name");
                if (namee.getLength() == 0) {
                    // not
                    continue;
                }
                String name = namee.item(0).getTextContent();
                NodeList m = ce.getElementsByTagName("members");
                List<String> members = new ArrayList<String>();
                for (int i = 0; i < m.getLength(); i++) {
                    Node mm = m.item(i);
                    members.add(mm.getTextContent());
                }
                PackageTypeMembers pdi = new PackageTypeMembers();
                pdi.setName(name);
                pdi.setMembers(members.toArray(new String[members.size()]));
                pd.add(pdi);
            }
        }
        _package r = new _package();
        r.setTypes(pd.toArray(new PackageTypeMembers[pd.size()]));
        r.setVersion(API_VERSION + "");
        return r;
    } catch (ParserConfigurationException pce) {
        throw new Exception("Cannot create XML parser", pce);
    }
}

```

```

    } catch (IOException ioe) {
        throw new Exception(ioe);
    } catch (SAXException se) {
        throw new Exception(se);
    }
}

/**
 * The login call is used to obtain a token from Salesforce.
 * This token must be passed to all other calls to provide
 * authentication.
 */
private boolean login() throws ServiceException {
    String userName = getUserInput("Enter username: ");
    String password = getUserInput("Enter password: ");
    /** Next, the sample client application initializes the binding stub.
     *
     * This is our main interface to the API for the Enterprise WSDL.
     * The getSoap method takes an optional parameter,
     * (a java.net.URL) which is the endpoint.
     * For the login call, the parameter always starts with
     * http(s)://login.salesforce.com. After logging in, the sample
     * client application changes the endpoint to the one specified
     * in the returned loginResult object.
     */
    binding = (SoapBindingStub) new SforceServiceLocator().getSoap();

    // Time out after a minute
    binding.setTimeout(60000);
    // Log in using the Enterprise WSDL binding
    LoginResult loginResult;
    try {
        System.out.println("LOGGING IN NOW...");
        loginResult = binding.login(userName, password);
    }
    catch (LoginFault ex) {

        // The LoginFault derives from AxisFault
        ExceptionCode exCode = ex.getExceptionCode();
        if (exCode == ExceptionCode.FUNCTIONALITY_NOT_ENABLED ||
            exCode == ExceptionCode.INVALID_CLIENT ||
            exCode == ExceptionCode.INVALID_LOGIN ||
            exCode == ExceptionCode.LOGIN_DURING_RESTRICTED_DOMAIN ||
            exCode == ExceptionCode.LOGIN_DURING_RESTRICTED_TIME ||
            exCode == ExceptionCode.ORG_LOCKED ||
            exCode == ExceptionCode.PASSWORD_LOCKOUT ||
            exCode == ExceptionCode.SERVER_UNAVAILABLE ||
            exCode == ExceptionCode.TRIAL_EXPIRED ||
            exCode == ExceptionCode.UNSUPPORTED_CLIENT) {
            System.out.println("Please be sure that you have a valid username " +
                "and password.");
        } else {
            // Write the fault code to the console
            System.out.println(ex.getExceptionCode());
            // Write the fault message to the console
            System.out.println("An unexpected error has occurred." + ex.getMessage());
        }
        return false;
    } catch (Exception ex) {
        System.out.println("An unexpected error has occurred: " + ex.getMessage());
        ex.printStackTrace();
        return false;
    }
    // Check if the password has expired
    if (loginResult.isPasswordExpired()) {
        System.out.println("An error has occurred. Your password has expired.");
        return false;
    }

    /** Once the client application has logged in successfully, we use

```

```

* the results of the login call to reset the endpoint of the service
* to the virtual server instance that is servicing your organization.
* To do this, the client application sets the ENDPOINT_ADDRESS_PROPERTY
* of the binding object using the URL returned from the LoginResult. We
* use the metadata binding from this point forward as we are invoking
* calls in the metadata WSDL.
*/
metadatabinding = (MetadataBindingStub)
    new MetadataServiceLocator().getMetadata();
metadatabinding._setProperty(MetadataBindingStub.ENDPOINT_ADDRESS_PROPERTY,
    loginResult.getMetadataServerUrl());

/** The sample client application now has an instance of the MetadataBindingStub
* that is pointing to the correct endpoint. Next, the sample client application
* sets a persistent SOAP header (to be included on all subsequent calls that
* are made with the SoapBindingStub) that contains the valid sessionId
* for our login credentials. To do this, the sample client application
* creates a new SessionHeader object and set its sessionId property to the
* sessionId property from the LoginResult object.
*/
// Create a new session header object and add the session id
// from the login return object
SessionHeader sh = new SessionHeader();
sh.setSessionId(loginResult.getSessionId());
/** Next, the sample client application calls the setHeader method of the
* SoapBindingStub to add the header to all subsequent method calls. This
* header will persist until the binding is destroyed or until the header
* is explicitly removed. The "SessionHeader" parameter is the name of the
* header to be added.
*/
// set the session header for subsequent call authentication
metadatabinding.setHeader(
    new MetadataServiceLocator().getServiceName().getNamespaceURI(),
    "SessionHeader", sh);

// return true to indicate that we are logged in, pointed
// at the right url and have our security token in place.
return true;
}

//The sample client application retrieves the user's login credentials.
// Helper function for retrieving user input from the console
String getUserInput(String prompt) {
    System.out.print(prompt);
    try {
        return rdr.readLine();
    }
    catch (IOException ex) {
        return null;
    }
}
}
}

```

RetrieveRequest

The RetrieveRequest object specified in a `retrieve()` call consists of the following properties:

Name	Type	Description
apiVersion	double	Required. The API version for the retrieve request. The API version determines the fields retrieved for each metadata type. For example, an <code>icon</code> field was added to the <code>CustomTab</code> for API version 14.0. If you retrieve

Name	Type	Description
		components for version 13.0 or earlier, the components will not include the <code>icon</code> field.
<code>packageNames</code>	<code>string[]</code>	A list of package names to be retrieved. If you are retrieving only unpackaged components, do not specify a name here. You can retrieve packaged and unpackaged components in the same retrieve.
<code>singlePackage</code>	<code>boolean</code>	Specifies whether only a single package is being retrieved (<code>true</code>) or not (<code>false</code>). If <code>false</code> , then more than one package is being retrieved.
<code>specificFiles</code>	<code>string[]</code>	A list of file names to be retrieved. If a value is specified for this property, <code>packageNames</code> must be set to <code>null</code> and <code>singlePackage</code> must be set to <code>true</code> .
<code>unpackaged</code>	Package	A list of components to retrieve that are not in a package.

checkRetrieveStatus ()

Checks the status of declarative metadata call `retrieve ()` and returns the zip file contents.

Syntax

```
RetrieveResult = metadatabinding.checkRetrieveStatus(ID id);
```

Usage

`checkRetrieveStatus` is part of the procedure for retrieving metadata components from an organization. It is used together with the `checkStatus` call which indicates when the asynchronous `retrieve` call has completed. Once `checkStatus` indicates that the call is completed, call `checkRetrieveStatus` to get the zip file contents:

1. Issue a `retrieve ()` call to start the asynchronous retrieval. An [AsyncResult](#) object is returned. If the call is completed, the `done` field contains `true`. Most often, the call is not completed quickly enough to be noted in the result. If it is completed, note the value in the `id` field returned and skip the next step.
2. If the call is not complete, issue a `checkStatus ()` call in a loop using the value in the `id` field of the [AsyncResult](#) object, returned by the `retrieve ()` call in the previous step. Check the [AsyncResult](#) object returned until the `done` field contains `true`. The time taken to complete a `retrieve ()` call depends on the size of the zip file being deployed, so use a longer wait time between iterations as the size of the zip file increases.
3. Issue a `checkRetrieveStatus ()` call to obtain the results of the `retrieve ()` call, using the `id` value returned in the first step.

Sample Code—Java

See the [retrieve \(\) sample code](#) for sample usage of this call.

Arguments

Name	Type	Description
<code>id</code>	ID	ID obtained from a RetrieveResult object returned by a <code>retrieve ()</code> call or a subsequent AsyncResult object returned by a <code>checkStatus ()</code> call.

Response

RetrieveResult

Chapter 7

CRUD-Based Calls

Use the following [CRUD-based](#) calls to work with metadata components in a manner similar to how synchronous API calls in the enterprise WSDL act upon objects.

create ()

Adds one or more new metadata components to your organization asynchronously.

This call can be used to create any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#) on page 75.

Syntax

```
AsyncResult[] = metadatabinding.create(Metadata[] metadata);
```

Usage

Use this call to add one or more metadata components to your organization.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Required Fields

Required fields are determined by the metadata components being created. For more information about specific component types, see [Metadata Components and Types](#) on page 75.

Valid Data Values

You must supply values that are valid for the field’s data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool (your development tool handles the appropriate mapping of data types in SOAP messages).

String Values

When storing values in string fields, the API trims any leading and trailing whitespace. For example, if the value of a [label](#) field is entered as "MyObject " the value is stored in the database as "MyObject".

Basic Steps for Creating Metadata Components

Use the following process to create metadata components:

1. Design an array and populate it with the components you want to create. All components must be of the same type.
2. Call `create ()` with the component array passed in as an argument.

3. An [AsyncResult](#) object is returned for each component you try to create, and is updated with status information as the operation moves from a queue to completed or error state. Call `checkStatus ()` in a loop until the status values in [AsyncResult](#) indicate that all create operations are completed. Start with a wait time of one second between iterations of `checkStatus ()` calls, and double the wait time each time you make a subsequent call.

Sample Code—Java

See [Step 3: Walk Through the Java Sample Code](#) on page 7 for sample Java code using the `create ()` call.

Arguments

Name	Type	Description
metadata	Metadata []	<p>Array of one or more metadata components.</p> <p>Limit: 10.</p> <p>You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.</p>

Response

[AsyncResult](#)[]

delete ()

Deletes one or more components from your organization asynchronously.

You can use this call to delete any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#) on page 75.

Syntax

```
AsyncResult [] = metadataConnection.delete (Metadata [] metadata);
```

Usage

Use this call to delete one or more components from your organization.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Rules and Guidelines

When deleting components, consider the following rules and guidelines:

- Your client application must be logged in with sufficient access rights to delete individual components within the specified component. For more information, see “Factors that Affect Data Access” in the [SOAP API Developer's Guide](#).
- In addition, you might also need permission to access this component’s parent component.
- To ensure referential integrity, this call supports cascading deletions. If you delete a parent component, you delete its children automatically, as long as each child component can be deleted.

Basic Steps for Deleting Metadata Components

Use the following process to delete metadata components:

1. Determine the `fullName` of each component you want to delete. See [Metadata](#) for more details on the `fullName` field. You can only delete components of the same type in a single call.
2. Invoke the `delete()` call, passing in the array of metadata components with `fullName` specified.
3. An `AsyncResult` object is returned for each component you try to delete, and is updated with status information as the operation moves from a queue to completed or error state. Call `checkStatus()` in a loop until the status values in `AsyncResult` indicate that all the delete operations are completed. Start with a wait time of one second between iterations of `checkStatus()` calls, and double the wait time each time you make a subsequent call.

Sample Code—Java

```
public void deleteCustomObject() {
    try {
        CustomObject co = new CustomObject();
        co.setFullName("MyCustomObject__c");
        AsyncResult[] ars = metadataConnection.create(new Metadata[]
            {co});
        AsyncResult asyncResult = ars[0];
        long waitTimeMilliSecs = 1000;
        while (!asyncResult.isDone()) {
            Thread.sleep(waitTimeMilliSecs);
            // double the wait time for the next iteration
            waitTimeMilliSecs *= 2;
            asyncResult = mdConnection.checkStatus(
                new String[] {asyncResult.getId()})[0];
            System.out.println("Status is: " + asyncResult.getState());
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    } catch (InterruptedException ie) {
        ie.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
metadata	Metadata []	<p>Array of one or more metadata components. You only need to set the <code>fullName</code> field in the Metadata object.</p> <p>Limit: 10.</p> <p>You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.</p>

Response

[AsyncResult](#)[]

update ()

Updates one or more components in your organization asynchronously.

This call can be used to update any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#) on page 75.

Syntax

```
AsyncResult[] = metadataConnection.update(UpdateMetadata[] metadata);
```

Usage

Use this call to update one or more components. This call is analogous to the `ALTER TABLE` statement in SQL.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Required Fields

You must supply values for all the required fields in the component.

Valid Field Values

You must supply values that are valid for the field’s data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool (your development tool handles the appropriate mapping of data types in SOAP messages).

String Values

When storing values in string fields, the API trims any leading and trailing white space. For example, if the value of a `label` field is entered as `"MyObject "` the value is stored in the database as `"MyObject"`.

Basic Steps for Updating Metadata Components

Use this process to update metadata components:

1. Create an array of `UpdateMetadata` components and populate it with the components you wish to update. All components must be of the same type.
2. Invoke the `update()` call, passing in the array of metadata components to update.
3. An `AsyncResult` object is returned for each component you try to update, and is updated with status information as the operation moves from a queue to completed or error state. In a loop, call `checkStatus()` until the status values in `AsyncResult` indicate that all the update operations are completed. Start with a wait time of one second between iterations of `checkStatus()` calls, and double the wait time each time you make a subsequent call.

Sample Code—Java

```
public void updateCustomObject() {
    try {
        CustomObject co = new CustomObject();
        String name = "MyCustomObject";
        co.setFullName(name + "_c");
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Created by the Metadata API");
        co.setEnableActivities(true);
        co.setLabel(name + " Object");
        co.setPluralLabel(co.getLabel() + "s");
        co.setSharingModel(SharingModel.ReadWrite);

        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setLabel(co.getFullName() + " Name");

        co.setNameField(nf);

        UpdateMetadata updateMetadata = new UpdateMetadata();
        updateMetadata.setMetadata(co);
        updateMetadata.setCurrentName("TheCurrentName");

        AsyncResult[] ars = metadataConnection.update(new UpdateMetadata[]
            { updateMetadata });
        AsyncResult asyncResult = ars[0];
    }
}
```

```

// set initial wait time to one second in milliseconds
long waitTimeMilliSecs = 1000;
while (!asyncResult.isDone()) {
    Thread.sleep(waitTimeMilliSecs);
    // double the wait time for the next iteration
    waitTimeMilliSecs *= 2;
    asyncResult = metadataConnection.checkStatus(
        new String[] {asyncResult.getId()}[0]);
    System.out.println("Status is: " + asyncResult.getState());
}

if (asyncResult.getState() != AsyncRequestState.Completed) {
    System.out.println(asyncResult.getStatusCode() + " msg: " +
        asyncResult.getMessage());
}
} catch (InterruptedException ie) {
    ie.printStackTrace();
} catch (ConnectionException ce) {
    ce.printStackTrace();
}
}
}

```

Arguments

Name	Type	Description
metadata	UpdateMetadata[]	<p>Array of one or more UpdateMetadata data structures that represent the components you wish to update.</p> <p>Limit: 10.</p> <p>You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.</p>

UpdateMetadata

One or more [UpdateMetadata](#) objects are defined in the `metadata` argument. This object can be used to update any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#) on page 75. Each [UpdateMetadata](#) object has the following fields:

Field	Field Type	Description
currentName	string	The API name of the component or field before the update. For example, if you wanted to update a CustomObject named Foo, the value of this field would be <code>FOO__c</code> . This value is supplied because this call may change the name, and the value here provides mapping.
metadata	Metadata	Full specification of the component or field you wish to update.

Response

[AsyncResult\[\]](#)

createMetadata ()

Adds one or more new metadata components to your organization synchronously.

Syntax

```
SaveResult[] = metadatabinding.createMetadata(Metadata[] metadata);
```

Usage

Use the `createMetadata()` call to add one or more metadata components to your organization. This call executes synchronously, meaning the call returns only when the operation completes.

This call can be used to create any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#).

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Required Fields

Required fields are determined by the metadata components being created. For more information about specific component types, see [Metadata Components and Types](#).

Valid Data Values

You must supply values that are valid for the field’s data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool (your development tool handles the appropriate mapping of data types in SOAP messages).

String Values

When storing values in string fields, the API trims any leading and trailing whitespace. For example, if the value of a `label` field is entered as “MyObject ”, the value is stored in the database as “MyObject”.

Basic Steps for Creating Metadata Components

Use the following process to create metadata components:

1. Design an array and populate it with the components that you want to create. All components must be of the same type.
2. Call `createMetadata()` with the component array passed in as an argument.
3. A `SaveResult` object is returned for each component you tried to create. It contains information about whether the operation was successful, the name of the component created, and any errors returned if the operation wasn’t successful.

Sample Code—Java

```
public void createCustomObjectSync() {
    try {
        CustomObject co = new CustomObject();
        String name = "MyCustomObject1";
        co.setFullName(name + "_c");
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Created by the Metadata API");
        co.setEnableActivities(true);
        co.setLabel(name + " Object");
        co.setPluralLabel(co.getLabel() + "s");
        co.setSharingModel(SharingModel.ReadWrite);

        CustomField nf = new CustomField();
```

```

nf.setType(FieldType.Text);
nf.setLabel(co.getFullName() + " Name");
co.setNameField(nf);

SaveResult[] results = metadataConnection
    .createMetadata(new Metadata[] { co });

for (SaveResult r : results) {
    if (r.isSuccess()) {
        System.out.println("Created component: " + r.getFullName());
    } else {
        System.out
            .println("Errors were encountered while creating "
                + r.getFullName());
        for (Error e : r.getErrors()) {
            System.out.println("Error message: " + e.getMessage());
            System.out.println("Status code: " + e.getStatusCode());
        }
    }
}
} catch (ConnectionException ce) {
    ce.printStackTrace();
}
}

```

Arguments

Name	Type	Description
metadata	Metadata []	<p>Array of one or more metadata components.</p> <p>Limit: 10.</p> <p>You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.</p>

Response

[SaveResult](#)[]

readMetadata ()

Returns one or more metadata components from your organization synchronously.

Syntax

```
ReadResult = metadataConnection.readMetadata(string metadataType, string[] fullNames);
```

Usage

Use the `readMetadata ()` call to get one or more metadata components from your organization. This call executes synchronously, meaning the call returns only when the operation completes.

You can use this call to retrieve any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#).

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Basic Steps for Reading Metadata Components

Use the following process to read metadata components:

1. Determine the metadata type of the components you want to read, and the `fullName` of each component to read. See [Metadata](#) for more details on the `fullName` field. You can read only components of the same type in a single call.
2. Invoke the `readMetadata()` call. For the first argument, pass in the name of the metadata type. The metadata type must match one of the values returned by the `describeMetadata()` call. For the second argument, pass in an array of full names corresponding to the components you wish to get. The full names must match one or more full names returned by the `listMetadata()` call.
3. A `ReadResult` is returned that contains an array of `Metadata` components. Cast each returned `Metadata` object to the metadata type you specified in the call to get the component’s properties.

Sample Code—Java

```
public void readCustomObjectSync() {
    try {
        ReadResult readResult = metadataConnection
            .readMetadata("CustomObject", new String[] {
                "MyCustomObject1__c", "MyCustomObject2__c" });
        Metadata[] mdInfo = readResult.getRecords();
        System.out.println("Number of component info returned: "
            + mdInfo.length);
        for (Metadata md : mdInfo) {
            if (md != null) {
                CustomObject obj = (CustomObject) md;
                System.out.println("Custom object full name: "
                    + obj.getFullName());
                System.out.println("Label: " + obj.getLabel());
                System.out.println("Number of custom fields: "
                    + obj.getFields().length);
                System.out.println("Sharing model: "
                    + obj.getSharingModel());
            } else {
                System.out.println("Empty metadata.");
            }
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
<code>metadataType</code>	<code>string</code>	The metadata type of the components to read.
<code>fullNames</code>	<code>string[]</code>	<p>Array of full names of the components to read.</p> <p>Limit: 10.</p> <p>You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.</p>

Response

[ReadResult](#)

updateMetadata ()

Updates one or more metadata components in your organization synchronously.

Syntax

```
SaveResult[] = metadataConnection.updateMetadata(Metadata[] metadata);
```

Usage

Use the `updateMetadata ()` call to update one or more metadata components in your organization. This call executes synchronously, meaning the call returns only when the operation completes.

You can use this call to update any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#).

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Required Fields

You must supply values for all the required fields in the component.

Valid Field Values

You must supply values that are valid for the field’s data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool (your development tool handles the appropriate mapping of data types in SOAP messages).

String Values

When storing values in string fields, the API trims any leading and trailing white space. For example, if the value of a `label` field is entered as `"MyObject "` the value is stored in the database as `"MyObject"`.

Basic Steps for Updating Metadata Components

Use this process to update metadata components:

1. Create an array of the components you wish to update. All components must be of the same type.
2. Invoke the `updateMetadata ()` call, passing in the array of metadata components to update.

A `SaveResult` object is returned for each component you tried to update. It contains information about whether the operation was successful, the name of the component updated, and any errors returned if the operation wasn’t successful.

Sample Code—Java

```
public void updateCustomObjectSync () {
    try {
        CustomObject co = new CustomObject();
        String name = "MyCustomObject1";
        co.setFullName(name + "__c");
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Updated description");
    }
}
```



```

co.setLabel(name + " Object Update");
co.setPluralLabel(co.getLabel() + "s");
co.setSharingModel(SharingModel.ReadWrite);

// Name field with a type and label is required
CustomField cf = new CustomField();
cf.setType(FieldType.Text);
cf.setLabel(co.getFullName() + " Name");
co.setNameField(cf);

SaveResult[] results = metadataConnection
    .updateMetadata(new Metadata[] { co });

for (SaveResult r : results) {
    if (r.isSuccess()) {
        System.out.println("Updated component: " + r.getFullName());
    } else {
        System.out
            .println("Errors were encountered while updating "
                + r.getFullName());
        for (Error e : r.getErrors()) {
            System.out.println("Error message: " + e.getMessage());
            System.out.println("Status code: " + e.getStatusCode());
        }
    }
}
} catch (ConnectionException ce) {
    ce.printStackTrace();
}
}

```

Arguments

Name	Type	Description
metadata	Metadata[]	<p>Array of one or more metadata components you wish to update.</p> <p>Limit: 10.</p> <p>You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.</p>

Response

[SaveResult\[\]](#)

deleteMetadata ()

Deletes one or more metadata components from your organization synchronously.

Syntax

```
DeleteResult[] = metadataConnection.delete(string metadataType, string[] fullNames);
```

Usage

Use the `deleteMetadata ()` call to delete one or more metadata components from your organization. This call executes synchronously, meaning the call returns only when the operation completes.

You can use this call to delete any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#).

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Rules and Guidelines

When deleting components, consider the following rules and guidelines:

- Your client application must be logged in with sufficient access rights to delete individual components within the specified component. For more information, see “Factors that Affect Data Access” in the *SOAP API Developer's Guide*.
- In addition, you might also need permission to access this component’s parent component.
- To ensure referential integrity, this call supports cascading deletions. If you delete a parent component, you delete its children automatically, as long as each child component can be deleted.

Basic Steps for Deleting Metadata Components

Use the following process to delete metadata components:

1. Determine the metadata type of the components you want to delete and the `fullName` of each component to delete. You can delete only components of the same type in a single call. The full names must match one or more full names returned by the `listMetadata()` call. See [Metadata](#) for more details on the `fullName` field.
2. Invoke the `deleteMetadata()` call. For the first argument, pass in the name of the metadata type. For the second argument, pass in an array of full names corresponding to the components you wish to delete.

A `DeleteResult` object is returned for each component you try to delete. It contains information about whether the operation was successful, the name of the deleted component, and any errors returned if the operation wasn’t successful.

Sample Code—Java

```
public void deleteCustomObjectSync() {
    try {
        DeleteResult[] results = metadataConnection.deleteMetadata(
            "CustomObject", new String[] { "MyCustomObject1__c",
            "MyCustomObject2__c" });
        for (DeleteResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Deleted component: " + r.getFullName());
            } else {
                System.out
                    .println("Errors were encountered while deleting "
                        + r.getFullName());
                for (Error e : r.getErrors()) {
                    System.out.println("Error message: " + e.getMessage());
                    System.out.println("Status code: " + e.getStatusCode());
                }
            }
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
<code>metadataType</code>	string	The metadata type of the components to delete.

Name	Type	Description
fullNames	string[]	<p>Array of full names of the components to delete.</p> <p>Limit: 10.</p> <p>You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.</p>

Response

[DeleteResult\[\]](#)

renameMetadata ()

Renames a metadata component in your organization synchronously.

Syntax

```
SaveResult = metadataConnection.renameMetadata(string metadataType, String oldFullname, String newFullname);
```

Usage

Use the `renameMetadata ()` call to rename one metadata component in your organization. This call executes synchronously, meaning the call returns only when the operation completes.

You can use this call to rename any of the objects that extend [Metadata](#). For more details, see [Metadata Components and Types](#).

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Basic Steps for Renaming Metadata Components

Use the following process to rename a metadata component:

1. Determine the metadata type of the component you want to rename, its current full name, and the new full name. See [Metadata](#) for more details on the `fullName` field.
2. Invoke the `renameMetadata ()` call. For the first argument, pass in the name of the metadata type. Pass in the old full name as the second argument and the new full name as the last argument.

A `SaveResult` object is returned that contains information about whether the operation was successful, the name of the renamed component (which is the new name if the renaming was successful), and any errors returned if the operation wasn't successful.

Sample Code—Java

```
public void renameCustomObjectSync () {
    try {
        SaveResult[] results = metadataConnection.renameMetadata(
            "CustomObject", "MyCustomObject1__c", "MyCustomObject1New__c");
        for (SaveResult r : results) {
```

```
    if (r.isSuccess()) {
        System.out.println("Renamed component: " + r.getName());
    }
    else {
        System.out.println("Errors were encountered while renaming " + r.getName());
        for(Error e : r.getErrors()) {
            System.out.println("Error message: " + e.getMessage());
            System.out.println("Status code: " + e.getStatusCode());
        }
    }
}
} catch (ConnectionException ce) {
    ce.printStackTrace();
} catch (InterruptedException ie) {
    ie.printStackTrace();
}
}
```

Arguments

Name	Type	Description
metadataType	string	The metadata type of the components to rename.
oldFullName	string	The current component full name.
newFullName	string	The new component full name.

Response

[SaveResult](#)

Chapter 8

Utility Calls

Use the following utility calls to gather information that is useful for working with the [file-based](#) or [CRUD-based](#) calls.

- `checkStatus()`
- `describeMetadata()`
- `listMetadata()`

checkStatus()

Checks the status of asynchronous metadata calls `create()`, `update()`, or `delete()`, or the declarative metadata call `retrieve()`.



Note: In API version 29.0, Database.com improved the deployment status properties and removed the requirement to use `checkStatus()` after a `deploy()` call to get information about deployments. Database.com continues to support the use of `checkStatus()` when using `deploy()` with API version 28.0 or earlier. For more information, see [deploy\(\)](#) on page 25.

Syntax

```
AsyncResult[] = metadatabinding.checkStatus(ID[] ids);
```

Usage

Use this call to check whether or not an asynchronous metadata call or declarative metadata call has completed.

Sample Code—Java

See [Step 3: Walk Through the Java Sample Code](#) on page 7 for sample Java code using this call.

Arguments

Name	Type	Description
<code>ids</code>	<code>ID[]</code>	Array of one or more IDs. Each ID is returned in an AsyncResult and corresponds to a component being created, updated, deleted, deployed, or retrieved.

Response

`AsyncResult[]`

describeMetadata ()

This call retrieves the metadata which describes your organization. This information includes Apex classes and triggers, custom objects, custom fields on standard objects, tab sets that define an app, and many other components.

Syntax

```
DescribeMetadataResult[] = metadataConnection.describeMetadata(double apiVersion);
```

Arguments

Name	Type	Description
apiVersion	double	The API version for which you want metadata; for example, 30.0.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Sample Code—Java

```
public void describeMetadata() {
    try {
        double apiVersion = 21.0;
        // Assuming that the SOAP binding has already been established.
        DescribeMetadataResult res =
            metadataConnection.describeMetadata(apiVersion);
        StringBuffer sb = new StringBuffer();
        if (res != null && res.getMetadataObjects().length > 0) {
            for (DescribeMetadataObject obj : res.getMetadataObjects()) {
                sb.append("*****\n");
                sb.append("XMLName: " + obj.getXmlName() + "\n");
                sb.append("DirName: " + obj.getDirectoryName() + "\n");
                sb.append("Suffix: " + obj.getSuffix() + "\n");
                sb.append("*****\n");
            }
        } else {
            sb.append("Failed to obtain metadata types.");
        }
        System.out.println(sb.toString());
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Response

[DescribeMetadataResult](#)

listMetadata ()

This call retrieves property information about metadata components in your organization. Data is returned for the components that match the criteria specified in the `queries` parameter. The `queries` array can contain up to three [ListMetadataQuery](#) queries for each call. This call supports every metadata type: both top-level, such as [CustomObject](#) and [ApexClass](#), and child types, such as [CustomField](#).

Syntax

```
FileProperties[] = metadataConnection.listMetadata(ListMetadataQuery[] queries, double
asOfVersion);
```

Usage

This call is useful when you want to identify individual components in `package.xml` for a `retrieve()` call or if you want a high-level view of particular metadata types in your organization. For example, you could use this call to return a list of names of all the `CustomObject` components in your organization, and use this information to make a subsequent `retrieve()` call to return a subset of these components. For more information about working with `package.xml`, see [Deploying and Retrieving Metadata](#) on page 14.



Note: This is a synchronous call so the results are returned in one call. This differs from asynchronous calls, such as `retrieve()`, where at least one subsequent call is needed to get the results.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Sample Code—Java

The sample code below lists information about your custom objects. The code assumes that the SOAP binding has already been established.

```
public void listMetadata() {
    try {
        ListMetadataQuery query = new ListMetadataQuery();
        query.setType("CustomObject");
        //query.setFolder(null);
        double asOfVersion = 30.0;
        // Assuming that the SOAP binding has already been established.
        FileProperties[] lmr = metadataConnection.listMetadata(
            new ListMetadataQuery[] {query}, asOfVersion);
        if (lmr != null) {
            for (FileProperties n : lmr) {
                System.out.println("Component fullName: " + n.getFullName());
                System.out.println("Component type: " + n.getType());
            }
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
queries	ListMetadataQuery[]	A list of objects that specify which components you are interested in.
asOfVersion	double	The API version for the metadata listing request. If you don't specify a value in this field, it defaults to the API version specified when you logged in. This field allows you to override the default and set another API version so that, for example, you could list the metadata for a metadata type that was added in a later version than the API version specified when you logged in. This field is available in API version 18.0 and later.

Response

[FileProperties](#)

ListMetadataQuery

The ListMetadataQuery parameter specified in a `listMetadata()` call consists of the following properties:

Name	Type	Description
folder	string	The folder associated with the component. This field is required for components that use folders.
type	string	Required. The metadata type, such as CustomObject, CustomField, or ApexClass.

Chapter 9

Result Objects

Use the following objects to get the results of your [file-based](#) or [CRUD-based](#) calls.

AsyncResult

Poll the values in this object to determine when an asynchronous metadata call has completed, and whether it was successful or not.

The asynchronous metadata calls [create\(\)](#), [update\(\)](#), and [delete\(\)](#) return an array of AsyncResult objects. Each element in the array corresponds to an element in the array of metadata components passed in the call.

Use the [checkStatus\(\)](#) call against each object to discover when the call is completed for that object. Database.com updates each AsyncResult object as the call completes, or when any errors occur.

The [retrieve\(\)](#) call uses AsyncResult similarly, though you must subsequently use [checkDeployStatus\(\)](#) or [checkRetrieveStatus\(\)](#) respectively to get more status information for the deployment or retrieval.

In API version 29.0, Database.com moved several properties from the AsyncResult object to the [DeployResult](#) on page 62 object, and added several new ones, to improve the process for getting information about deployments. For more information about these changes, see [deploy\(\)](#) on page 25.

For API versions 29.0 and later, the AsyncResult object has the following properties.

Name	Type	Description
done	boolean	Required. Indicates whether the call has completed (<code>true</code>) or not (<code>false</code>).
id	ID	Required. ID of the component being created, updated, deleted, deployed, or retrieved.
message	string	Message corresponding to the <code>statusCode</code> field returned, if any.
state	AsyncRequestState (enumeration of type string)	Required. The AsyncRequestState object has one of four possible values: <ul style="list-style-type: none">Queued: This call has not started. It is waiting in a queue.InProgress: This call has started, but has not completed yet.Completed: This call has completed.Error: An error occurred. See the <code>statusCode</code> for more information.

Name	Type	Description
statusCode	StatusCode (enumeration of type string)	If an error occurred during the <code>create()</code> , <code>update()</code> , or <code>delete()</code> call, a status code is returned, and the message corresponding to the status code is returned in the <code>message</code> field. For a description of each StatusCode value, see “StatusCode” in the <i>SOAP API Developer's Guide</i> .

For API versions 28.0 and earlier, the AsyncResult object has the following properties.

Name	Type	Description
checkOnly	boolean	Indicates whether this deployment is being used to check the validity of the deployed files without making any changes in the organization (<code>true</code>) or not (<code>false</code>). A check-only deployment does not deploy any components or change the organization in any way. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
done	boolean	Required. Indicates whether the call has completed (<code>true</code>) or not (<code>false</code>).
id	ID	Required. ID of the component being created, updated, deleted, deployed, or retrieved.
message	string	Message corresponding to the <code>statusCode</code> field returned, if any.
numberComponentErrors	int	The number of components that generated errors during this deployment. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
numberComponentsDeployed	int	The number of components that have been deployed so far for this deployment. This field in conjunction with the <code>numberComponentsTotal</code> field gives you an indication of the progress of the deployment. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
numberComponentsTotal	int	The total number of components in the deployment. This field in conjunction with the <code>numberComponentsDeployed</code> field gives you an indication of the progress of the deployment. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
numberTestErrors	int	The number of Apex tests that have generated errors during this deployment. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
numberTestsCompleted	int	The number of Apex tests that have completed so far for this deployment. This field in conjunction with the <code>numberTestsTotal</code> field gives you an indication of the progress of tests for the deployment. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
numberTestsTotal	int	The total number of Apex tests in the deployment. This field in conjunction with the <code>numberTestsCompleted</code> field gives you an indication of the progress of tests for the deployment. The value in this field is not accurate until the deployment has started running tests for the

Name	Type	Description
		components being deployed. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
<code>secondsToWait</code>	<code>int</code>	<p>This field is no longer supported for API version 13.0 and later and is only provided for backward compatibility. The field was removed in API version 17.0.</p> <p>Indicates the number of seconds before the call is likely to complete. This is an estimate only. A reasonable approach is to wait one second before calling <code>checkStatus()</code> to see if the operation is complete. Double your wait time for each successive iteration of <code>checkStatus()</code> calls until the operation is complete.</p>
<code>state</code>	<code>AsyncRequestState</code> (enumeration of type string)	<p>Required. The <code>AsyncRequestState</code> object has one of four possible values:</p> <ul style="list-style-type: none"> <code>Queued</code>: This call has not started. It is waiting in a queue. <code>InProgress</code>: This call has started, but has not completed yet. <code>Completed</code>: This call has completed. <code>Error</code>: An error occurred. See the <code>statusCode</code> for more information.
<code>stateDetail</code>	<code>string</code>	Indicates which component is currently being deployed or which Apex test class is running. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
<code>stateDetailLastModifiedDate</code>	<code>dateTime</code>	The data and time when the <code>stateDetail</code> field was last modified. This field is available in API version 16.0 and later and is only relevant for the <code>deploy()</code> call.
<code>statusCode</code>	<code>StatusCode</code> (enumeration of type string)	<p>If an error occurred during the <code>create()</code>, <code>update()</code>, <code>delete()</code> or <code>deploy()</code> call, a status code is returned, and the message corresponding to the status code is returned in the <code>message</code> field.</p> <p>For a description of each <code>StatusCode</code> value, see “<code>StatusCode</code>” in the <i>SOAP API Developer's Guide</i>.</p>

CancelDeployResult

Contains information about a deployment cancellation—whether the cancellation completed and the deployment ID.

The asynchronous metadata call `cancelDeploy()` returns a `CancelDeployResult` object.

Version

Available in API version 30.0 and later.

`CancelDeployResult` has the following properties.

Name	Type	Description
done	boolean	Indicates whether the deployment cancellation, which is started through <code>cancelDeploy()</code> , has completed (<code>true</code>) or not (<code>false</code>). When a deployment hasn't started yet and is still in the queue, the deployment is canceled immediately with the <code>cancelDeploy()</code> call and this field returns <code>true</code> . Otherwise, this field returns <code>false</code> when the cancellation is in progress.
id	ID	ID of the deployment being canceled.

DeployResult

Contains information about the success or failure of the associated `deploy()` call.

The asynchronous metadata call `checkDeployStatus()` returns a `DeployResult` object.

In API version 29.0, Database.com moved several properties from the `AsyncResult` on page 59 object to the `DeployResult` object to improve the process for getting information about deployments. For more information about these changes, see `deploy()` on page 25.

For API versions 29.0 and later, the `DeployResult` object has the following properties.

Name	Type	Description
id	ID	ID of the component being deployed.
canceledBy	ID	The ID of the user who canceled the deployment. This field is available in API version 30.0 and later.
canceledByName	string	The full name of the user who canceled the deployment. This field is available in API version 30.0 and later.
checkOnly	boolean	Indicates whether this deployment is being used to check the validity of the deployed files without making any changes in the organization (<code>true</code>) or not (<code>false</code>). A check-only deployment does not deploy any components or change the organization in any way.
completedDate	dateTime	Timestamp for when the deployment process ended.
createdBy	ID	The ID of the user who created the deployment. This field is available in API version 30.0 and later.
createdByName	string	The full name of the user who created the deployment. This field is available in API version 30.0 and later.
createdDate	dateTime	Timestamp for when the <code>deploy()</code> call was received.
details	DeployDetails []	Provides the details of a deployment that is in-progress or ended, if the <code>includeDetails</code> parameter is set to <code>true</code> in the <code>checkDeployStatus()</code> call.

Name	Type	Description
done	boolean	Indicates whether the server finished processing the <code>deploy()</code> call for the specified <code>id</code> .
errorMessage	string	Message corresponding to the values in the <code>errorStatusCode</code> field, if any.
errorStatusCode	string	If an error occurred during the <code>deploy()</code> call, a status code is returned, and the message corresponding to the status code is returned in the <code>errorMessage</code> field. For a description of each <code>StatusCode</code> value, see “ <code>StatusCode</code> ” in the SOAP API Developer's Guide .
ignoreWarnings	boolean	Optional. Defaults to <code>false</code> . Specifies whether a deployment should continue even if the deployment generates warnings. Do not set this argument to <code>true</code> for deployments to production organizations.
lastModifiedDate	dateTime	Timestamp of the last update for the deployment process.
numberComponentErrors	int	The number of components that generated errors during this deployment.
numberComponentsDeployed	int	The number of components deployed in the deployment process. Use this value with the <code>numberComponentsTotal</code> value to get an estimate of the deployment's progress.
numberComponentsTotal	int	The total number of components in the deployment. Use this value with the <code>numberComponentsDeployed</code> value to get an estimate of the deployment's progress.
numberTestErrors	int	The number of Apex tests that have generated errors during this deployment.
numberTestsCompleted	int	The number of completed Apex tests for this deployment. Use this value with the <code>numberTestsTotal</code> value to get an estimate of the deployment's test progress.
numberTestsTotal	int	The total number of Apex tests for this deployment. Use this value with the <code>numberTestsCompleted</code> value to get an estimate of the deployment's test progress. The value in this field is not accurate until the deployment has started running tests for the components being deployed.
runTestsEnabled	boolean	Indicates whether Apex tests were run as part of this deployment (<code>true</code>) or not (<code>false</code>). Tests are either automatically run as part of a deployment or can be set to run in <code>DeployOptions</code> for the <code>deploy()</code> call. For information on when tests are automatically run, see Running Tests in a Deployment . This field is available in API version 30.0 and later.
rollbackOnError	boolean	Optional. Defaults to <code>true</code> . Indicates whether any failure causes a complete rollback (<code>true</code>) or not (<code>false</code>). If <code>false</code> , 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 <code>true</code> if you are deploying to a production organization.
startDate	dateTime	Timestamp for when the deployment process began.

Name	Type	Description
stateDetail	string	Indicates which component is currently being deployed or which Apex test class is running.
status	DeployStatus (enumeration of type string)	Indicates the current state of the deployment. The valid values are: <ul style="list-style-type: none"> • Pending • InProgress • Succeeded • SucceededPartial • Failed • Canceling • Canceled
success	boolean	Indicates whether the deployment was successful (<code>true</code>) or not (<code>false</code>).

DeployDetails

These fields provide more information for the `details` field of the `DeployResult` object, if the `includeDetails` parameter is set to (`true` in the `deploy()` call).



Note: While a deployment is still in-progress, the `DeployDetails` object only contains `componentFailures` data. After the deployment process finishes, the other fields populate with the data for the entire deployment.

Name	Type	Description
componentFailures	DeployMessage []	One or more <code>DeployMessage</code> objects containing deployment errors for each component.
componentSuccesses	DeployMessage []	One or more <code>DeployMessage</code> objects containing successful deployment details for each component.
retrieveResult	RetrieveResult	If the <code>performRetrieve</code> parameter was specified for the <code>deploy()</code> call, a <code>retrieve()</code> call is performed immediately after the <code>deploy()</code> process completes. This field contains the results of that retrieval.
runTestResult	RunTestsResult	If the <code>runAllTests</code> or <code>runTests</code> parameters are set to run tests for the <code>deploy()</code> call, this field contains the results of those tests. While a deployment is still in-progress, this field only contains error data. After the deployment process finishes, this field populates with the data for the entire deployment.

For API versions 28.0 and earlier, the `DeployResult` object has the following properties.

Name	Type	Description
id	ID	ID of the component being deployed.
messages	DeployMessage []	Contains information about the success or failure of a <code>deploy()</code> call.
retrieveResult	RetrieveResult	If the <code>performRetrieve</code> parameter was specified for the <code>deploy()</code> call, a <code>retrieve()</code> call is performed immediately after the <code>deploy()</code> process completes. This field contains the results of that retrieval.
runTestResult	RunTestsResult	If the <code>runAllTests</code> or <code>runTests</code> parameters are set to run tests for the <code>deploy()</code> call, this field contains the results of those tests.

Name	Type	Description
success	boolean	Indicates whether the deployment was successful (<code>true</code>) or not (<code>false</code>).

DeployMessage

Each `DeployResult` object contains one or more `DeployMessage` objects. Each `DeployMessage` object contains information about the deployment success or failure of a component in the deployment `.zip` file:

Name	Type	Description
changed	boolean	If <code>true</code> , the component was changed as a result of this deployment. If <code>false</code> , the deployed component was the same as the corresponding component already in the organization.
columnNumber	int	Each component is represented by a text file. If an error occurred during deployment, this field represents the column of the text file where the error occurred.
componentType	string	The metadata type of the component in this deployment. This field is available in API version 30.0 and later.
created	boolean	If <code>true</code> , the component was created as a result of this deployment. If <code>false</code> , the component was either deleted or modified as a result of the deployment.
createdDate	dateTime	The date and time when the component was created as a result of this deployment. This field is available in API version 30.0 and later.
deleted	boolean	If <code>true</code> , the component was deleted as a result of this deployment. If <code>false</code> , the component was either new or modified as result of the deployment.
fileName	string	The name of the file in the <code>.zip</code> file used to deploy this component.
fullName	string	The full name of the component. Inherited from Metadata , this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.
id	ID	ID of the component being deployed.
lineNumber	int	Each component is represented by a text file. If an error occurred during deployment, this field represents the line number of the text file where the error occurred.
problem	string	If an error or warning occurred, this field contains a description of the problem that caused the compile to fail.
problemType	DeployProblemType (enumeration of type string)	Indicates the problem type. The problem details are tracked in the problem field. The valid values are: <ul style="list-style-type: none"> Warning Error This field is available in API version 18.0 and later. Prior to version 18.0, there was no distinction between warnings and errors. All problems were treated as errors and prevented a successful deployment.
success	boolean	Indicates whether the component was successfully deployed (<code>true</code>) or not (<code>false</code>).

RunTestsResult

The call returns information about whether or not the compilation of the specified Apex was successful and if the unit tests completed successfully.

A RunTestsResult object has the following properties

Name	Type	Description
codeCoverage	CodeCoverageResult []	An array of one or more CodeCoverageResult objects that contains the details of the code coverage for the specified unit tests.
codeCoverageWarnings	CodeCoverageWarning []	An array of one or more code coverage warnings for the test run. The results include both the total number of lines that could have been executed, as well as the number, line, and column positions of code that was not executed.
failures	RunTestFailure []	An array of one or more RunTestFailure objects that contain information about the unit test failures, if there are any.
numFailures	int	The number of failures for the unit tests.
numTestsRun	int	The number of unit tests that were run.
successes	RunTestSuccess []	An array of one or more RunTestSuccesses objects that contain information about successes, if there are any.
totalTime	double	The total cumulative time spent running tests. This can be helpful for performance monitoring.

CodeCoverageResult

The [RunTestsResult](#) object contains this object. It contains information about whether or not the compile of the specified Apex and run of the unit tests was successful.

Name	Type	Description
dmlInfo	CodeLocation []	For each class or trigger tested, for each portion of code tested, this property contains the DML statement locations, the number of times the code was executed, and the total cumulative time spent in these calls. This can be helpful for performance monitoring.
id	ID	The ID of the CodeLocation . The ID is unique within an organization.
locationsNotCovered	CodeLocation []	For each class or trigger tested, if any code is not covered, the line and column of the code not tested, and the number of times the code was executed.
methodInfo	CodeLocation []	For each class or trigger tested, the method invocation locations, the number of times the code was executed, and the total cumulative time spent in these calls. This can be helpful for performance monitoring.
name	string	The name of the class or trigger covered.

Name	Type	Description
namespace	string	The namespace that contained the unit tests, if one is specified.
numLocations	int	The total number of code locations.
soqlInfo	CodeLocation []	For each class or trigger tested, the location of SOQL statements in the code, the number of times this code was executed, and the total cumulative time spent in these calls. This can be helpful for performance monitoring.
type	string	Do not use. In early, unsupported releases, used to specify class.

CodeCoverageWarning

The [RunTestsResult](#) object contains this object. It contains information about the Apex class which generated warnings.

This object has the following properties.

Name	Type	Description
id	ID	The ID of the CodeLocation . The ID is unique within an organization.
message	string	The message of the warning generated.
name	string	The namespace that contained the unit tests, if one is specified.
namespace	string	The namespace that contained the unit tests, if one is specified.

RunTestFailure

The [RunTestsResult](#) object returns information about failures during the unit test run.

This object has the following properties.

Name	Type	Description
id	ID	The ID of the class which generated failures.
message	string	The failure message.
methodName	string	The name of the method that failed.
name	string	The name of the class that failed.
namespace	string	The namespace that contained the class, if one was specified.
stackTrace	string	The stack trace for the failure.
time	double	The time spent running tests for this failed operation. This can be helpful for performance monitoring.
type	string	Do not use. In early, unsupported releases, used to specify class.

RunTestSuccess

The [RunTestsResult](#) object returns information about successes during the unit test run.

This object has the following properties.

Name	Type	Description
id	ID	The ID of the class which generated the success.
methodName	string	The name of the method that succeeded.
name	string	The name of the class that succeeded.
namespace	string	The namespace that contained the unit tests, if one is specified.
time	double	The time spent running tests for this operation. This can be helpful for performance monitoring.

CodeLocation

The [RunTestsResult](#) object contains this object in a number of fields.

This object has the following properties.

Name	Type	Description
column	int	The column location of the Apex tested.
line	int	The line location of the Apex tested.
numExecutions	int	The number of times the Apex was executed in the test run.
time	double	The total cumulative time spent at this location. This can be helpful for performance monitoring.

DescribeMetadataResult

Contains information about the organization that is useful for developers working with declarative metadata.

The `describeMetadata()` call returns a DescribeMetadataResult object.

Each DescribeMetadataResult object has the following properties:

Name	Type	Description
metadataObjects	DescribeMetadataObject []	One or more metadata components and their attributes.
organizationNamespace	string	The namespace of the organization. Specify only for Developer Edition organizations that can contain a managed package. The managed package has a namespace specified when it is created.

Name	Type	Description
<code>partialSaveAllowed</code>	boolean	Indicates whether <code>rollbackOnError</code> is allowed (<code>true</code>) or not (<code>false</code>). This value is always : <ul style="list-style-type: none"> <code>false</code> in production organizations. the opposite of <code>testRequired</code>.
<code>testRequired</code>	boolean	Indicates whether tests are required (<code>true</code>) or not (<code>false</code>). This value is always the opposite of <code>partialSaveAllowed</code> .

DescribeMetadataObject

This object is returned as part of the `DescribeMetadataResult`. Each `DescribeMetadataObject` has the following properties:

Name	Type	Description
<code>childXmlNames</code>	string[]	List of child sub-components for this component.
<code>directoryName</code>	string	The name of the directory in the <code>.zip</code> file that contains this component.
<code>inFolder</code>	boolean	Indicates whether the component is in a folder (<code>true</code>) or not (<code>false</code>). For example, documents, email templates and reports are stored in folders.
<code>metaFile</code>	boolean	Indicates whether the component requires an accompanying metadata file. For example, documents, classes, and s-controls are components that require an additional metadata file.
<code>suffix</code>	string	The file suffix for this component.
<code>xmlName</code>	string	The name of the root element in the metadata file for this component. This name also appears in the <code>Packages > types > name</code> field in the manifest file <code>package.xml</code> .

ReadResult

Contains result information for the `readMetadata` call.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
<code>records</code>	Metadata[]	An array of metadata components returned from <code>readMetadata()</code> .

RetrieveResult

Contains information about the success or failure of the associated `retrieve()` call.

The metadata `retrieve()` call returns a `RetrieveResult` object.

Each RetrieveResult object has the following fields:

Name	Type	Description
fileProperties	FileProperties []	Contains information about the properties of each component in the .zip file, and the manifest file <code>package.xml</code> . One object per component is returned.
id	ID	ID of the component being retrieved.
messages	RetrieveMessage []	Contains information about the success or failure of the <code>retrieve()</code> call.
zipFile	base64Binary	The zip file returned by the retrieve request. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client.

FileProperties

This component contains information about the properties of each component in the .zip file, and the manifest file `package.xml`. One object per component is returned. Note that this component does not contain information about any associated metadata files in the .zip file, only the component files and manifest file. FileProperties contains the following properties:

Name	Type	Description
createdById	string	Required. ID of the user who created the file.
createdByName	string	Required. Name of the user who created the file.
createdDate	dateTime	Required. Date and time when the file was created.
fileName	string	Required. Name of the file.
fullName	string	Required. The file developer name used as a unique identifier for API access. The value is based on the <code>fileName</code> but the characters allowed are more restrictive. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
id	string	Required. ID of the file.
lastModifiedById	string	Required. ID of the user who last modified the file.
lastModifiedByName	string	Required. Name of the user who last modified the file.
lastModifiedDate	dateTime	Required. Date and time that the file was last modified.
manageableState	ManageableState (enumeration of type string)	Indicates the manageable state of the specified component if it is contained in a package: <ul style="list-style-type: none"> • beta • deleted • deprecated • installed • released • unmanaged <p>For more information about states of manageability for components in Force.com AppExchange packages, see “Planning the Release of Managed Packages” in the Database.com online help.</p>

Name	Type	Description
namespacePrefix	string	If any, the namespace prefix of the component.
type	string	Required. The metadata type, such as CustomObject, CustomField, or ApexClass.

RetrieveMessage

RetrieveResult returns this object, which contains information about the success or failure of the `retrieve()` call. One object per problem is returned:

Name	Type	Description
fileName	string	The name of the file in the retrieved .zip file where a problem occurred.
problem	string	A description of the problem that occurred.

SaveResult

Contains result information for the `createMetadata`, `updateMetadata`, or `renameMetadata` call.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
errors	Error[]	An array of errors returned if the operation wasn't successful.
fullName	string	The full name of the component processed.
success	boolean	Indicates whether the operation was successful (<code>true</code>) or not (<code>false</code>).

DeleteResult

Contains result information for the `deleteMetadata` call.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
errors	Error[]	An array of errors returned if the operation wasn't successful.
fullName	string	The full name of the deleted component.
success	boolean	Indicates whether the deletion was successful (<code>true</code>) or not (<code>false</code>).

Error

Represents an error that occurred during a synchronous CRUD (`createMetadata()`, `updateMetadata()`, or `deleteMetadata()`) operation.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
<code>fields</code>	<code>string[]</code>	An array containing names of fields that affected the error condition.
<code>message</code>	<code>string</code>	The error message text.
<code>statusCode</code>	<code>StatusCode</code>	A status code corresponding to the error. For a description of each <code>StatusCode</code> value, see “ <code>StatusCode</code> ” in the SOAP API Developer's Guide .

Chapter 10

Metadata Types

Metadata API doesn't allow you to access everything that you can customize in the user interface. The following table lists all of the metadata types that can be retrieved or deployed with Metadata API, whether or not the type can be retrieved with the wildcard (*) symbol in `package.xml`, and links to detail pages for each type. For more information about using wildcards, see [Working with the Zip File](#) on page 14. Metadata types don't always correspond directly to their related data types, so in some cases the information is accessible, but not organized as you might expect. For example, dependent picklists are exposed as a type of picklist, not a separate metadata type.

Metadata Type	Allows Wildcard (*)?	Description
ApexClass	Yes	Represents an Apex class. An Apex class is 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.
ApexTrigger	Yes	Represents an Apex trigger. A trigger is Apex code that executes before or after specific data manipulation language (DML) events occur, such as before object records are inserted into the database, or after records have been deleted.
BaseSharingRule	See entries for CriteriaBasedSharingRule and OwnerSharingRule .	Represents the base container for criteria-based and owner-based sharing rules.
BusinessHoursSettings on page 151	Yes	Represents the metadata used to manage settings for business hours and holidays in entitlements, entitlement templates, campaigns, and cases.
CallCenter	Yes	Represents the Call Center definition used to integrate Database.com with a third-party computer-telephony integration (CTI) system.
CriteriaBasedSharingRule	Yes, for all types except Custom Objects.	Represents a criteria-based sharing rule. CriteriaBasedSharingRule enables you to share records based on specific criteria.
CustomField	No	Represents the metadata associated with a custom field. Use this metadata type to create, update, or delete custom field definitions.
CustomObject	Yes, for Field Sets and Record Types, No for other components.	Represents a custom object that stores data unique to your organization.
Group	Yes	Represents a set of public groups, which can have users, roles, and other groups.
Layout	Yes	Represents the metadata associated with a page layout.

Metadata Type	Allows Wildcard (*)?	Description
Metadata	No	This is the base class for all metadata types. You cannot edit this object. A component is an instance of a metadata type.
MetadataWithContent	No	This is the base type for all metadata types that contain content, such as documents or email templates.
MobileSettings	Yes	Represents an organization's mobile settings, such as mobile Chatter settings, whether Mobile Lite is enabled for an organization, and so on.
NamedFilter	No	This component has been removed as of API version 30.0 and is only provided for backward compatibility. The metadata associated with a lookup filter is now represented by the <code>lookupFilter</code> field in the CustomField component. Represents the metadata associated with a lookup filter. Use this metadata type to create, update, or delete lookup filter definitions.
OwnerSharingRule	Yes, for all types except Custom Objects.	Represents an ownership-based sharing rule. OwnerSharingRule enables you to share records owned by a set of users with another set, using rules that specify the access level of the target user group.
Package	No	Used to specify metadata components to be retrieved as part of a <code>retrieve()</code> call, or to define a package of components.
PermissionSet	Yes	Represents a set of permissions that's used to grant additional access to one or more users without changing their profile or reassigning profiles. You can use permission sets to grant access, but not to deny access.
Picklist (Including Dependent Picklist)	No	Represents a picklist (or dependent picklist) definition for a custom field in a custom object or a custom or standard field in a standard object.
Profile	Yes	Represents a user profile. A profile defines a user's permission to perform different functions within Database.com.
Queue	Yes	Represents a holding area for items before they are processed.
RemoteSiteSetting	Yes	Represents a remote site setting.
Role	Yes	Represents a role in your organization.
SecuritySettings	Yes	Represents an organization's security settings. Security settings define trusted IP ranges for network access, password and login requirements, and session expiration and security settings.
SharingReason	No	Represents an Apex sharing reason, which is used to indicate why sharing was implemented for a custom object.
SharingRecalculation	No	Represents Apex classes that recalculate the Apex managed sharing for a specific custom object.
SharingRules	See entries for CriteriaBasedSharingRule and OwnerSharingRule .	Represents a set of sharing rules. SharingRules enables you to share records with a set of users, using rules that specify the access level of the target user group.
ValidationRule	No	Represents a validation rule, which is used to verify that the data a user enters in a record is valid and can be saved. A validation rule contains a formula or expression that evaluates the data in one or more fields and returns a value of <code>true</code> or <code>false</code> . Validation rules also include an error message that your client application can display to the user when the rule returns a value of <code>true</code> due to invalid data.

Metadata Type	Allows Wildcard (*)?	Description
Workflow	Yes	Represents the metadata associated with a 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.

Metadata Components and Types

Metadata components are not based on sObjects, like objects in the API. Instead, they are based on metadata types, such as `ApexClass` and `CustomObject`, which extend [Metadata](#). A component is an instance of a metadata type. For example, `CustomObject` is a metadata type for custom objects, and the `MyCustomObject__c` component is an instance of a custom object.

A metadata type can be identified in the metadata WSDL as any `complexType` that extends the [Metadata](#) `complexType`. A `complexType` that is a metadata type includes the following element in its WSDL definition:

```
<xsd:extension base="tns:Metadata">
```

`CustomObject` and `BusinessProcess` extend `Metadata` so they are metadata types; `ActionOverride` doesn't extend `Metadata` so it's not a metadata type.

You can individually deploy or retrieve a component for a metadata type. For example, you can retrieve an individual `BusinessProcess` component, but you can't retrieve an individual `ActionOverride` component. You can only retrieve an `ActionOverride` component by retrieving its encompassing `CustomObject` component.

Metadata components can be manipulated by [asynchronous Metadata API calls](#) or [declarative \(or file-based\) Metadata API calls](#).

Most of the components can be accessed using Force.com IDE. Exceptions are noted in the description of the object.

Field Data Types

Each component field has a specific field type. These field types can correspond to other components defined in the WSDL, or primitive data types, like `string`, that are commonly used in strongly typed programming languages.

These field data types are used in the SOAP messages that are exchanged between your client application and the API. When writing your client application, follow the data typing rules defined for your programming language and development environment. Your development tool handles the mapping of typed data in your programming language with these SOAP data types.

For more information about primitive data types, see the [SOAP API Developer's Guide](#).

Enumeration Fields

Some component fields have a data type that is an enumeration. An enumeration is the API equivalent of a picklist. The valid values of the field are restricted to a strict set of possible values, all having the same data type. These values are listed in the field description column for each enumeration field. The XML below shows a sample definition of an enumeration of type string in the WSDL.

```
<xsd:simpleType name="DashboardComponentFilter">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="RowLabelAscending"/>
    <xsd:enumeration value="RowLabelDescending"/>
    <xsd:enumeration value="RowValueAscending"/>
    <xsd:enumeration value="RowValueDescending"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
</xsd:restriction>
</xsd:simpleType>
```

Supported Calls

All of the metadata types are supported by the main calls, unless it is stated otherwise in the individual component sections. The main Metadata API calls are `create()`, `delete()`, `update()`, `deploy()`, `retrieve()`, `listMetadata()`, and `describeMetadata()`. All other calls, such as `checkStatus()`, are considered utility calls as they are used in conjunction with one of the main calls.

Unsupported Metadata Types

Some things you can customize in a Database.com organization aren't available in Metadata API.

The following components can't be retrieved or deployed with Metadata API, and changes to them must be made manually in each of your organizations:

- Currency Exchange Rates
- Delegated Administration
- Sharing Organization Wide Defaults
- User Interface Settings (except calendar features, which are supported in [ActivitiesSettings](#) on page 148)

ApexClass

Represents an Apex class. An Apex class is 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. For more information, see the *Force.com Apex Code Developer's Guide*. This metadata type extends the `MetadataWithContent` component and shares its fields.



Note: You can't deploy updates to an Apex class if there are one or more active scheduled jobs for that class.

Supported Calls

`deploy()`, `retrieve()`, `describeMetadata()`, `listMetadata()`



Note: This metadata type is not supported by the `create()`, `delete()`, and `update()` calls.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.cls` for the class file. The accompanying metadata file is named `ClassName-meta.xml`.


Apex classes are stored in the `classes` folder in the corresponding package directory.

Version

Apex classes are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
apiVersion	double	The API version for this class. Every class has an API version specified at creation.
content	base64	The Apex class definition. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field is inherited from the MetadataWithContent component.
fullName	string	The Apex class name. The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. This field is inherited from the Metadata component.
status	ApexCodeUnitStatus (enumeration of type string)	<p>The current status of the Apex class. The following string values are valid:</p> <ul style="list-style-type: none"> Active - The class is active. Deleted - The class is marked for deletion. This is useful for managed packages, because it allows a class to be deleted when a managed package is updated. <p> Note: ApexCodeUnitStatus includes an <code>Inactive</code> option, but it is only supported for ApexTrigger; it is not supported for <code>ApexClass</code>.</p>

Declarative Metadata Sample Definition

The following sample creates the `MyHelloWorld.cls` class, and the corresponding `MyHelloWorld.cls-meta.xml` metadata file.

MyHelloWorld.cls file:

```
public class MyHelloWorld {
    // This method updates the Hello field on a list
    // of accounts.
    public static void addHelloWorld(Account[] accs){
        for (Account a:accs){
            if (a.Hello__c != 'World')
                a.Hello__c = 'World';
        }
    }
}
```

MyHelloWorld.cls-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexClass xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>30.0</apiVersion>
</ApexClass>
```

See Also:

[ApexTrigger](#)

ApexTrigger

Represents an Apex trigger. A trigger is Apex code that executes before or after specific data manipulation language (DML) events occur, such as before object records are inserted into the database, or after records have been deleted. For more information, see “Managing Apex Triggers” in the Database.com online help. This metadata type extends the [MetadataWithContent](#) component and shares its fields.

Supported Calls

`deploy()`, `retrieve()`, `describeMetadata()`, `listMetadata()`



Note: This metadata type is not supported by the `create()`, `delete()`, and `update()` calls.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.trigger` for the trigger file. The accompanying metadata file is named `TriggerName-meta.xml`.

Apex triggers are stored in the `triggers` folder in the corresponding package directory.

Version

Triggers are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
<code>apiVersion</code>	double	Required. The API version for this trigger. Every trigger has an API version specified at creation.
<code>content</code>	base64	The Apex trigger definition. This field is inherited from the MetadataWithContent component.
<code>fullName</code>	string	The Apex trigger name. The name can only contain characters, letters, and the underscore (<code>_</code>) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. This field is inherited from the Metadata component.
<code>status</code>	ApexCodeUnitStatus (enumeration of type string)	Required. The current status of the Apex trigger. The following string values are valid: <ul style="list-style-type: none"> <code>Active</code> - The trigger is active. <code>Inactive</code> - The trigger is inactive, but not deleted. <code>Deleted</code> - The trigger is marked for deletion. This is useful for managed packages, because it allows a trigger to be deleted when a managed package is updated.

Declarative Metadata Sample Definition

The following sample creates the `MyHelloWorld.trigger` trigger, and the corresponding `MyHelloWorld.trigger-meta.xml` metadata file.

MyHelloWorld.trigger file:

```
trigger helloWorldAccountTrigger on Account (before insert) {
    Account[] accs = Trigger.new;
    MyHelloWorld.addHelloWorld(accs);
}
```

MyHelloWorld.trigger-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexTrigger xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>30.0</apiVersion>
</ApexTrigger>
```

See Also:

[ApexClass](#)

AppMenu

Represents the Force.com app menu or the Salesforce1 navigation menu.

File Suffix and Directory Location

Each AppMenu component gets stored in a single file in the folder of the corresponding package directory. The filename uses the format *Feature.appMenu*.

- There's one app switcher app menu file stored in a file named `AppSwitcher.appMenu`.
- There's one Salesforce1 app menu file stored in a file named `Salesforce1.appMenu`.

These two files are located in the `appMenus` folder. The `.appMenu` files are different from other named components, as there's only one file for each AppMenu component. App menu files can't be created or deleted.

Version

AppMenu components are available in API version 30.0 and later.

Fields

Field Name	Field Type	Description
<code>appMenuItems</code>	AppMenuItem []	A list of menu items in the app menu.

AppMenuItem

Represents a menu item in the app menu.

Field Name	Field Type	Description
<code>name</code>	string	The API name of the item.
<code>type</code>	string	The type of application represented by this item. Acceptable values for <code>AppSwitcher.appMenu</code> are: <ul style="list-style-type: none"> • <code>ConnectedApp</code>

Field Name	Field Type	Description
		<ul style="list-style-type: none"> • CustomApplication • ServiceProvider <p>Acceptable values for Salesforce1.appMenu are:</p> <ul style="list-style-type: none"> • CustomApplication • CustomTab • StandardAppMenuItem. <p>The name for this item can be:</p> <ul style="list-style-type: none"> ◇ MyDay ◇ Feed ◇ Tasks ◇ Dashboards ◇ Search ◇ People (available only when Chatter is enabled) ◇ Groups (available only when Chatter is enabled)

Declarative Metadata Sample Definition

The following is an example of an AppSwitcher.appMenu file.

```
<?xml version="1.0" encoding="UTF-8"?>
<AppMenu xmlns="http://soap.sforce.com/2006/04/metadata">
  <appMenuItems>
    <appMenuItem>
      <name>standard_Sales</name>
      <type>CustomApplication</type>
    </appMenuItem>
    <appMenuItem>
      <name>standard_Support</name>
      <type>CustomApplication</type>
    </appMenuItem>
    <appMenuItem>
      <name>CustomApp1</name>
      <type>CustomApplication</type>
    </appMenuItem>
    <appMenuItem>
      <name>CustomApp2</name>
      <type>CustomApplication</type>
    </appMenuItem>
    <appMenuItem>
      <name>ConnectedApp1</name>
      <type>ConnectedApp</type>
    </appMenuItem>
  </appMenuItems>
</AppMenu>
```

The following is an example package.xml that references the previous definition.

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

The following is an example of a `Salesforce1.appMenu` component.

```
<?xml version="1.0" encoding="UTF-8"?>
<AppMenu xmlns="http://soap.sforce.com/2006/04/metadata">
  <appMenuItems>
    <appMenuItem>
      <name>StandardItem1</name>
      <type>StandardItem</type>
    </appMenuItem>
    <appMenuItem>
      <name>StandardItem2</name>
      <type>StandardItem</type>
    </appMenuItem>
    <appMenuItem>
      <name>StandardItem3</name>
      <type>StandardItem</type>
    </appMenuItem>
    <appMenuItem>
      <name>CustomTab1</name>
      <type>CustomTab</type>
    </appMenuItem>
  </appMenuItems>
</AppMenu>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>Salesforce1</members>
    <name>AppMenu</name>
  </types>
```

The following is an example of a package manifest used to deploy or retrieve all the available app menu metadata for an organization, using a wildcard:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>*</members>
    <name>AppMenu</name>
  </types>
  <version>30.0</version>
</Package>
```

Usage

Use `AppSwitcher.appMenu` to reorder the list of menu items that appears in the Force.com app menu. You can't add app menu items to or remove app menu items from `AppSwitcher.appMenu`.

Use `Salesforce1.appMenu` to customize the list of menu items that appears in the Salesforce1 navigation menu by reordering, adding, or removing the app menu items.

CallCenter

Represents the Call Center definition used to integrate Database.com with a third-party computer-telephony integration (CTI) system.

File Suffix and Directory Location

CallCenter components have the suffix `callCenter` and are stored in the `callCenters` folder.

Version

CallCenter components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
adapterUrl	string	Optional field. A URL that points to a CTI 4 adapter.
displayName	string	The display name of this call center.
displayNameLabel	string	The label of the <code>displayName</code> field in Call Center setup page.
internalNameLabel	string	The label of the <code>internalName</code> field in Call Center setup page.
version	string	The version of this call center.
sections	CallCenterSection[]	Custom setup items defined for this call center.

CallCenterSection

Field Name	Field Type	Description
items	CallCenterItem[] on page 82	Contains the label, name, and value that describe the sections.
label	string	The label of the section.
name	string	The name of the section.

CallCenterItem

Field Name	Field Type	Description
label	string	The label of the custom setup item.
name	string	The name of the custom setup item.
value	int or URL	The value of the custom setup item.

Declarative Metadata Sample Definition

The following is an example of a CallCenter component:

```
<?xml version="1.0" encoding="UTF-8"?>
<CallCenter xmlns="http://soap.sforce.com/2006/04/metadata">
  <adapterUrl>http://localhost:11000</adapterUrl>
  <displayName>Demo Call Center Adapter</displayName>
  <displayNameLabel>Display Name</displayNameLabel>
  <internalNameLabel>Internal Name</internalNameLabel>
  <sections>
    <items>
      <label>Description</label>
      <name>reqDescription</name>
      <value>Demo Call Center Adapter</value>
    </items>
    <items>
      <label>CTI Connector ProgId</label>
      <name>reqProgId</name>
      <value>DemoAdapter.DemoAdapter.1</value>
    </items>
  </sections>
</CallCenter>
```



```

        <label>Version</label>
        <name>reqVersion</name>
        <value>3.0</value>
    </items>
    <items>
        <label>CTI Adapter URL</label>
        <name>reqAdapterUrl</name>
        <value>http://localhost:11000</value>
    </items>
    <label>General Information</label>
    <name>reqGeneralInfo</name>
</sections>
<sections>
    <items>
        <label>Outside Prefix</label>
        <name>reqOutsidePrefix</name>
        <value>1</value>
    </items>
    <items>
        <label>Long Distance Prefix</label>
        <name>reqLongDistPrefix</name>
        <value>1</value>
    </items>
    <items>
        <label>International Prefix</label>
        <name>reqInternationalPrefix</name>
        <value>01</value>
    </items>
    <label>Dialing Options</label>
    <name>reqDialingOptions</name>
</sections>
<version>4</version>
</CallCenter>

```

CustomObject

Represents a custom object that stores data unique to your organization. It extends the [Metadata](#) metadata type and inherits its `fullName` field. You must specify all relevant fields when you create or update a custom object. You cannot update a single field on the object. For more information about custom objects, see “Custom Object Record Overview” in the Database.com online help.

All metadata components have a `fullName` field, which must be fully specified for any custom object.

For example, the following are fully specified names:

```

Account
MyCustomObject__c

```

For sample Java code that creates a custom object, see [Step 3: Walk Through the Java Sample Code](#) on page 7.

Declarative Metadata File Suffix and Directory Location

Custom object names are automatically appended with `__c`. The file suffix is `.object` for the custom object (or standard object) file.

Custom and standard objects are stored in the `objects` folder in the corresponding package directory.



Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.


Version

Custom objects are available in API version 10.0 and later.

Fields

Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
customSettingsType	CustomSettingsType (enumeration of type string)	<p>When this field is present, this component is not a custom object, but a custom setting. This field returns the type of custom setting. The following string values are valid:</p> <ul style="list-style-type: none"> <code>List</code>—static data stored in cache and accessed as part of your application and available organization-wide. <code>Hierarchy</code>—static data stored in cache and accessed as part of your application and available based on a hierarchy of user, profile or organization. This is the default value. <p>This field is available in API version 17.0 and later.</p>
customSettingsVisibility	CustomSettingsVisibility (enumeration of type string)	<p>When this field is present, this component is not a custom object, but a custom setting. This field returns the visibility of the custom setting. The following string values are valid:</p> <ul style="list-style-type: none"> <code>Public</code>—if the custom setting is packaged, it is accessible to all subscribing organizations. <code>Protected</code>—if the custom setting is in a managed package, it is only accessible to the developer organization. Subscribing organizations cannot access it. This is the default value. <p>This field is available in API version 17.0 and later.</p>
deploymentStatus	DeploymentStatus (enumeration of type string)	Indicates the deployment status of the custom object.
deprecated	boolean	Reserved for future use.
description	string	A description of the object. Maximum of 1000 characters.
enableActivities	boolean	Indicates whether the custom object is enabled for activities (<code>true</code>) or not (<code>false</code>).
enableDivisions	boolean	Indicates whether the custom object is enabled for divisions (<code>true</code>) or not (<code>false</code>). For more information about the Division object, see the SOAP API Developer's Guide .
enableEnhancedLookup	boolean	Indicates whether the custom object is enabled for enhanced lookups (<code>true</code>) or not (<code>false</code>). Enhanced lookups provide an updated lookup dialog interface that gives users the ability to filter, sort, and page through search results as well as customize search result columns. For more information about enhanced lookups, see “Enabling Enhanced Lookups” in the Database.com online help.

Field Name	Field Type	Description
enableFeeds	boolean	Indicates whether the custom object is enabled for feed tracking (<code>true</code>) or not (<code>false</code>). For more information, see “Customizing Chatter Feed Tracking” in the Database.com online help. This field is available in API version 18.0 and later.
enableHistory	boolean	Indicates whether the custom object is enabled for history tracking (<code>true</code>) or not (<code>false</code>). Also available for standard objects in API version 29.0 and later.
fields	CustomField[]	Represents one or more fields in the object.
fullName	string	Inherited from Metadata , this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See <code>create()</code> to see an example of this field specified for a call. This value cannot be <code>null</code> .
gender	Gender	Indicates the gender of the noun. This is used for languages where words need different treatment depending on their gender.
label	string	Label that represents the object throughout the Database.com user interface.
namedFilter	NamedFilter[]	Represents the metadata associated with a lookup filter. Use this metadata type to create, update, or delete lookup filter definitions. This field is available in API version 17.0 and later. This field has been removed as of API version 30.0 and is only available in prior versions. The metadata associated with a lookup filter is now represented by the <code>lookupFilter</code> field in the <code>CustomField</code> component.
nameField	CustomField	Required. The field that this object's name is stored in. Every custom object must have a name, usually a string or autonumber. Identifier for the custom object record. This name appears in related lists and lookup dialogs. Every custom object must have a name, usually a string or autonumber.
pluralLabel	string	Plural version of the <code>label</code> value.
sharingModel	SharingModel	Indicates the sharing model for this custom object. Valid values are: <ul style="list-style-type: none"> • <code>Private</code> • <code>Read</code> • <code>ReadWrite</code>  Note: Using API version 29.0 and earlier, this field is read-only and you can't set this field through Metadata API; you must use the Database.com user interface. Using API version 30.0 and later, you can set this field for internal

Field Name	Field Type	Description
		users using the API and the Database.com user interface.
sharingReasons	SharingReason []	The reasons why the custom object is being shared.
sharingRecalculations	SharingRecalculation []	A list of custom sharing recalculations associated with the custom object.
startsWith	StartsWith (enumeration of type string)	Indicates whether the noun starts with a vowel, consonant, or is a special character. This is used for languages where words need different treatment depending on the first character. Valid values are listed in StartsWith .
validationRules	ValidationRule []	An array of one or more validation rules on this object.
webLinks	Weblink []	An array of one or more weblinks defined for the object.

Declarative Metadata Additional Components

CustomObject definitions may include additional components which are defined in the custom object for declarative metadata. The following components are defined in the CustomObject:

- [CustomField](#)
- [SharingReason](#)
- [SharingRecalculation](#)
- [ValidationRule](#)
- [Weblink](#)

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  <deploymentStatus>Deployed</deploymentStatus>
  <description>just a test object with one field for eclipse ide testing</description>
  <fields>
    <fullName>Comments__c</fullName>
    <description>add your comments about this object here</description>
    <inlineHelpText>This field contains comments made about this object</inlineHelpText>

    <label>Comments</label>
    <length>32000</length>
    <type>LongTextArea</type>
    <visibleLines>30</visibleLines>
  </fields>
  <label>MyFirstObject</label>
  <nameField>
    <label>MyFirstObject Name</label>
    <type>Text</type>
  </nameField>
  <pluralLabel>MyFirstObjects</pluralLabel>
</CustomObject>
```

```
<sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

See Also:

[CustomField](#)

[Metadata](#)

[Picklist \(Including Dependent Picklist\)](#)

[Weblink](#)

CustomField

Represents the metadata associated with a custom field. Use this metadata type to create, update, or delete custom field definitions. It extends the [Metadata](#) metadata type and inherits its `fullName` field. You can also use this metadata type to work with customizations of standard picklist fields, such as the `Industry` field for accounts.

You must specify the full name whenever you create or update a custom field. For example, a custom field on a custom object:

```
MyCustomObject__c.MyCustomField__c
```

Another example, a custom field on a standard object:

```
Account.MyAcctCustomField__c
```

Declarative Metadata File Suffix and Directory Location

Custom fields are defined as part of the custom object or standard object definition. See [CustomObject](#) for more information.



Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Retrieving Custom Fields on Custom or Standard Objects

When you retrieve a custom or standard object, you return everything associated with the object. However, you can also retrieve only the custom fields for an object by explicitly naming the object and fields in `package.xml`. The following definition in `package.xml` will create the files `objects/MyCustomObject__c.object` and `objects/Account.object__c.object`, each containing one custom field definition.

```
<types>
  <members>MyCustomObject__c.MyCustomField__c</members>
  <members>Account.MyCustomAccountField__c</members>
  <name>CustomField</name>
</types>
```

Version

Custom fields are available in API version 10.0 and later.

Fields

Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
caseSensitive	boolean	Indicates whether the field is case sensitive (<code>true</code>) or not (<code>false</code>).
defaultValue	string	If specified, represents the default value of the field.
deleteConstraint	DeleteConstraint (enumeration of type string)	Provides deletion options for lookup relationships. Valid values are: SetNull This is the default. If the lookup record is deleted, the lookup field is cleared. Restrict Prevents the record from being deleted if it's in a lookup relationship. Cascade Deletes the lookup record as well as associated lookup fields. For more information on lookup relationships, see "Overview of Object Relationships" in the Database.com Help.
deprecated	boolean	Reserved for future use.
description	string	Description of the field.
displayFormat	string	The display format.
displayLocationInDecimal	boolean	Indicates how the value of a Geolocation custom field appears in the user interface. If <code>true</code> , the geolocation appears in decimal notation. If <code>false</code> , the geolocation appears as degrees, minutes, and seconds.
externalId	boolean	Indicates whether the field is an external ID field (<code>true</code>) or not (<code>false</code>).
formula	string	If specified, represents a formula on the field.
formulaTreatBlankAs	TreatBlanksAs (enumeration of type string)	Indicates how to treat blanks in a formula. Valid values are <code>BlankAsBlank</code> and <code>BlankAsZero</code> .
fullName	string	Inherited from Metadata , this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See <code>create()</code> to see an example of this field specified for a call. This value cannot be <code>null</code> .
indexed	boolean	Indicates if the field is indexed. If this field is unique or the <code>externalId</code> is set <code>true</code> , the <code>isIndexed</code> value is set to <code>true</code> . This field has been deprecated as of version 14.0 and is only provided for backward compatibility.
inlineHelpText	string	Represents the content of field-level help. For more information, see "Defining Field-Level Help" in the Database.com Help.

Field Name	Field Type	Description
reparentableMasterDetail	boolean	Indicates whether the child records in a master-detail relationship on a custom object can be reparented to different parent records. The default value is <code>false</code> . This field is available in API version 25.0 and later.
label	string	Label for the field. You cannot update the label for standard picklist fields, such as the <code>Industry</code> field for accounts.
length	int	Length of the field.
lookupFilter	LookupFilter	Represents the metadata associated with a lookup filter. Use this metadata type to create, update, or delete lookup filter definitions. This field is available in API version 30.0 and later.
maskChar	EncryptedFieldMaskChar (enumeration of type string)	For encrypted fields, specifies the character to be used as a mask. Valid values are enumerated in EncryptedFieldMaskChar . For more information on encrypted fields, see “About Encrypted Custom Fields” in the Database.com Help.
maskType	EncryptedFieldMaskType (enumeration of type string)	For encrypted text fields, specifies the format of the masked and unmasked characters in the field. Valid values are enumerated in EncryptedFieldMaskType . For more information on encrypted fields, see “About Encrypted Custom Fields” in the Database.com Help.
picklist	Picklist	If specified, the field is a picklist, and this field enumerates the picklist values and labels.
populateExistingRows	boolean	Indicates whether existing rows will be populated (<code>true</code>) or not (<code>false</code>).
precision	int	The precision for number values. Precision is the number of digits in a number. For example, the number 256.99 has a precision of 5.
referenceTo	string	If specified, indicates a reference this field has to another object.
relationshipLabel	string	Label for the relationship.
relationshipName	string	If specified, indicates the value for one-to-many relationships. For example, in the object <code>MyObject</code> that had a relationship to <code>YourObject</code> , the relationship name might be <code>YourObjects</code> .
relationshipOrder	int	This field is valid for all master-detail relationships, but the value is only non-zero for junction objects. A junction object has two master-detail relationships, and is analogous to an association table in a many-to-many relationship. Junction objects must define one parent object as primary (0), the other as secondary (1). The definition of primary or secondary affects delete behavior and inheritance of look

Field Name	Field Type	Description
		and feel, and record ownership for junction objects. For more information, see the Database.com Help. 0 or 1 are the only valid values, and 0 is always the value for objects that are not junction objects.
required	boolean	Indicates whether the field requires a value on creation (<code>true</code>) or not (<code>false</code>).
scale	int	The scale for the field. Scale is the number of digits to the right of the decimal point in a number. For example, the number 256.99 has a scale of 2.
startingNumber	int	If specified, indicates the starting number for the field.
stripMarkup	boolean	Set to <code>true</code> to remove markup, or <code>false</code> to preserve markup. Used when converting a rich text area to a long text area.
summarizedField	string	Represents the field on the detail row that is being summarized. This field cannot be null unless the summaryOperation value is <code>count</code> .
summaryFilterItems	FilterItem[]	Represents the set of filter conditions for this field if it is a summary field. This field will be summed on the child if the filter conditions are met.
summaryForeignKey	string	Represents the master-detail field on the child that defines the relationship between the parent and the child.
summaryOperation	SummaryOperations (enumeration of type string)	Represents the sum operation to be performed. Valid values are enumerated in SummaryOperations .
trackFeedHistory	boolean	Indicates whether the field is enabled for feed tracking (<code>true</code>) or not (<code>false</code>). To set this field to <code>true</code> , the enableFeeds field on the associated CustomObject must also be <code>true</code> . For more information, see “Customizing Chatter Feed Tracking” in the Database.com Help. This field is available in API version 18.0 and later.
trackHistory	boolean	Indicates whether history tracking is enabled for the field (<code>true</code>) or not (<code>false</code>). Also available for standard object fields (picklist and lookup fields only) in API version 30.0 and later. To set <code>trackHistory</code> to <code>true</code> , the <code>enableHistory</code> field on the associated standard or custom object must also be <code>true</code> . For more information, see “Tracking Field History” in the Database.com Help.
trackTrending	boolean	Indicates whether historical trending data is captured for the field (<code>true</code>) or not (<code>false</code>). An object is enabled for historical trending if this attribute is <code>true</code> for at least one field. Available in API version 29.0 and later.

Field Name	Field Type	Description
		For more information, see “Report on Historical Trends” in the Database.com Help.
trueValueIndexed	boolean	This is only relevant for a checkbox field. If set, true values are built into the index. This field has been deprecated as of API version 14.0 and is only provided for backward compatibility.
type	FieldType	Required. Indicates the field type for the field. Valid values are enumerated in FieldType .
unique	boolean	Indicates whether the field is unique (<code>true</code>) or not (<code>false</code>).
visibleLines	int	Indicates the number of lines displayed for the field.
writeRequiresMasterRead	boolean	<p>Sets the minimum sharing access level required on the master record to create, edit, or delete child records. This field applies only to master-detail or junction object custom field types.</p> <ul style="list-style-type: none"> <code>true</code>—Allows users with “Read” access to the master record permission to create, edit, or delete child records. This setting makes sharing less restrictive. <code>false</code>—Allows users with “Read/Write” access to the master record permission to create, edit, or delete child records. This setting is more restrictive than <code>true</code>, and is the default value. <p>For junction objects, the most restrictive access from the two parents is enforced. For example, if you set to <code>true</code> on both master-detail fields, but users have “Read” access to one master record and “Read/Write” access to the other master record, users won't be able to create, edit, or delete child records.</p>

Custom fields use additional data types. For more information, see [Metadata Field Types](#) on page 107.

EncryptedFieldMaskChar

This field type is used in [maskChar](#). It is a string with two valid values: `asterisk` or `x`. For more information on encrypted fields, see About Encrypted Custom Fields in the Database.com online help.

EncryptedFieldMaskType

This field type is used in [maskType](#). Valid values are:

all

All characters in the field are hidden. This option is equivalent to the `Mask All Characters` option in Database.com.

creditCard

The first 12 characters are hidden and the last four display. This option is equivalent to the `Credit Card Number` option in Database.com.

ssn

The first five characters are hidden and the last four display. This option is equivalent to the `Social Security Number` option in Database.com.

lastFour

All characters are hidden but the last four display. This option is equivalent to the `Last Four Characters Clear` option in Database.com.

sin

All characters are hidden but the last four display. This option is equivalent to the `Social Insurance Number` option in Database.com.

nino

All characters are hidden. Database.com automatically inserts spaces after each pair of characters if the field contains nine characters. This option is equivalent to the `National Insurance Number` option in Database.com.

For more information on encrypted fields, see [About Encrypted Custom Fields](#) in the Database.com online help.

LookupFilter

Represents the metadata associated with a lookup filter. Replaces the `NamedFilter` component, which was removed as of API version 30.0. `LookupFilter` is available in API version 30.0 and later.

Field	Field Type	Description
<code>active</code>	boolean	Required. Indicates whether or not the lookup filter is active.
<code>booleanFilter</code>	string	Specifies advanced filter conditions. For more information on advanced filter conditions, see “Getting the Most Out of Filter Logic” in the Database.com Help.
<code>description</code>	string	A description of what this filter does.
<code>errorMessage</code>	string	The error message that appears if the lookup filter fails.
<code>filterItems</code>	FilterItem[]	Required. The set of filter conditions. You can have up to 10 <code>FilterItems</code> per lookup filter.
<code>infoMessage</code>	string	The information message displayed on the page. Use to describe things the user might not understand, such as why certain items are excluded in the lookup filter.
<code>isOptional</code>	boolean	Required. Indicates whether or not the lookup filter is optional.

Lookup filters use additional data types. For more information, see [Metadata Field Types](#).

FilterItem

Represents one entry in a set of filter criteria.

Field	Field Type	Description
<code>field</code>	string	Represents the field specified in the filter.
<code>operation</code>	FilterOperation (enumeration of type string)	Represents the filter operation for this filter item. Valid values are enumerated in FilterOperation .

Field	Field Type	Description
value	string	Represents the value of the filter item being operated upon, for example, if the filter is <code>my_number_field__c > 1</code> , the value of value is 1.
valueField	string	Specifies if the final column in the filter contains a field or a field value.

FilterOperation

This is an enumeration of type string that lists different filter operations. Valid values are:

- equals
- notEqual
- lessThan
- greaterThan
- lessOrEqual
- greaterOrEqual
- contains
- notContain
- startsWith
- includes
- excludes

SummaryOperations

Represents the type of a [summaryOperation](#). Valid values are:

- Count
- Min
- Max
- Sum

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
<fields>
  <fullName>Comments__c</fullName>
  <description>add your comments about this object here</description>
  <inlineHelpText>This field contains comments made about this object</inlineHelpText>

  <label>Comments</label>
  <length>32000</length>
  <type>LongTextArea</type>
  <visibleLines>30</visibleLines>
</fields>
```

```
.....
</CustomObject>
```

See Also:

[CustomObject](#)

[Picklist \(Including Dependent Picklist\)](#)

[Metadata](#)

[NamedFilter](#)

NamedFilter



Note: This component has been removed as of API version 30.0 and is only available in previous API versions. The metadata associated with a lookup filter is now represented by the `lookupFilter` field in the `CustomField` component.

Represents the metadata associated with a lookup filter. Use this metadata type to create, update, or delete lookup filter definitions. It extends the [Metadata](#) metadata type and inherits its `fullName` field. You can also use this metadata type to work with customizations of lookup filters on standard fields.



Note: The `namedFilter` appears as a child of the target object of the associated lookup field.

Declarative Metadata File Suffix and Directory Location

Lookup filters are defined as part of the custom object or standard object definition. See [CustomObject](#) for more information.



Note: Retrieving a component of this metadata type in a project makes the component appear in any `Profile` and `PermissionSet` components that are retrieved in the same package.


Version

Lookup filters are available in API version 17.0 and later.

Fields

Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
<code>active</code>	boolean	Required. Indicates whether or not the lookup filter is active.
<code>booleanFilter</code>	string	Specifies advanced filter conditions. For more information on advanced filter conditions, see “Getting the Most Out of Filter Logic” in the Database.com online help.
<code>description</code>	string	A description of what this filter does.
<code>errorMessage</code>	string	The error message that appears if the lookup filter fails.

Field Name	Field Type	Description
field	string	Required. The <code>fullName</code> of the custom or standard field associated with the lookup filter. You can associate one relationship field with each lookup filter, and vice-versa.  Note: You cannot update a field associated with a lookup filter.
filterItems	FilterItems[]	Required. The set of filter conditions.
infoMessage	string	The information message displayed on the page. Use to describe things the user might not understand, such as why certain items are excluded in the lookup filter.
fullName	string	Inherited from Metadata , this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See <code>create()</code> to see an example of this field specified for a call. This value cannot be <code>null</code> .
isOptional	boolean	Required. Indicates whether or not the lookup filter is optional.
name	string	Required. The name of the lookup filter. If you create this field in the user interface, a name is automatically assigned. If you create this field through Metadata API, you must include the <code>name</code> field.
sourceObject	string	The object that contains the lookup field that uses this lookup filter. Set this field if the lookup filter references fields on the source object.

Lookup filters use additional data types. For more information, see [Metadata Field Types](#).

FilterItems

FilterItems contains the following properties:

Field	Field Type	Description
field	string	Represents the field specified in the filter.
operation	FilterOperation (enumeration of type string)	Represents the filter operation for this filter item. Valid values are enumerated in FilterOperation .
value	string	Represents the value of the filter item being operated upon, for example, if the filter is <code>my_number_field__c > 1</code> , the value of <code>value</code> is 1.

FilterOperation

This is an enumeration of type string that lists different filter operations. Valid values are:

- `equals`
- `notEqual`

- lessThan
- greaterThan
- lessOrEqual
- greaterOrEqual
- contains
- notContain
- startsWith
- includes
- excludes

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
  <namedfilters>
    <fullName>nf_Acc</fullName>
    <active>true</active>
    <booleanFilter>1 OR 2</booleanFilter>
    <field>Account.lk__c</field>
    <filterItems>
      <field>Account.Phone</field>
      <operation>notEqual</operation>
      <value>x</value>
    </filterItems>
    <filterItems>
      <field>Account.Fax</field>
      <operation>notEqual</operation>
      <value>y</value>
    </filterItems>
    <name>Acc</name>
    <sourceObject>Account</sourceObject>
  </namedfilters>
....
</CustomObject>
```

See Also:

[CustomObject](#)

[Picklist \(Including Dependent Picklist\)](#)

[Metadata](#)

[CustomField](#)

Picklist (Including Dependent Picklist)

Represents a picklist (or dependent picklist) definition for a custom field in a custom object or a custom or standard field in a standard object.

Version

Picklists for custom fields in custom objects are available in API version 12.0 and later. Picklists for custom or standard fields in standard objects are available in API version 16.0 and later. Picklist values are deleted if necessary on a deploy of a custom field in API version 27.0 and later.

Declarative Metadata File Suffix and Directory Location

Picklist definitions are included in the custom object and field with which they are associated.

Fields

Picklist contains the following fields:


Field Name	Field Type	Description
controllingField	string	The <code>fullName</code> of the controlling field if this is a dependent picklist. A dependent picklist works in conjunction with a controlling picklist or checkbox to filter the available options. The value chosen in the controlling field affects the values available in the dependent field. This field is available in API version 14.0 and later.
picklistValues	PicklistValue[]	Required. Represents a set of values for a picklist.
sorted	boolean	Required. Indicates whether values should be sorted (<code>true</code>), or not (<code>false</code>).

PicklistValue

This metadata type defines a value in the picklist and specifies whether this value is the default value. It extends the [Metadata](#) metadata type and inherits its `fullName` field. Note the following when working with picklist values:

- When you retrieve a standard object, you all picklist values are retrieved, not just the customized picklist values.
- When you deploy changes to standard picklist fields, picklist values are added, updated or deleted as needed.
- You can't set a picklist value as inactive, but if the picklist value is missing and you invoke an `update()` call, the missing value becomes inactive.

Field Name	Field Type	Description
allowEmail	boolean	Indicates whether this value lets users email a quote PDF (<code>true</code>), or not (<code>false</code>). This field is only relevant for the <code>Status</code> field in quotes. This field is available in API version 18.0 and later.
closed	boolean	Indicates whether this value is associated with a closed status (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Status</code> field in cases and tasks. This field is available in API version 16.0 and later.
color	string	Indicates the color assigned to the picklist value when used in charts on reports and dashboards. The color is in hexadecimal format; for example <code>#FF6600</code> . If a color is not specified, it will be assigned dynamically on chart generation. This field is available in API version 17.0 and later.
controllingFieldValues	string[]	A list of values in the controlling field that are linked to this picklist value. The controlling field can be a checkbox or a picklist. This field is available in API version 14.0 and later. The values in the list depend on the field type: <ul style="list-style-type: none"> • Checkbox: checked or unchecked. • Picklist: The <code>fullName</code> of the picklist value in the controlling field.
converted	boolean	Indicates whether this value is associated with a converted status (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Lead Status</code> field in leads. Your organization can set its own guidelines for determining when a lead is qualified, but typically, you want to convert a lead as soon as it becomes a real opportunity

Field Name	Field Type	Description
		that you want to forecast. For more information, see “Converting Leads” in the Database.com online help. This field is available in API version 16.0 and later.
cssExposed	boolean	<p>Indicates whether this value is available in your Self-Service Portal (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Case Reason</code> field in cases.</p> <p>Self-Service provides an online support channel for your customers - allowing them to resolve their inquiries without contacting a customer service representative. For more information about Self-Service, see “Setting Up Self-Service” in the Database.com online help.</p> <p> Note: Starting with Spring '12, the Self-Service portal isn't available for new organizations. Existing organizations continue to have access to the Self-Service portal.</p> <p>This field is available in API version 16.0 and later.</p>
default	boolean	Required. Indicates whether this value is the default picklist value in the specified picklist (<code>true</code>), or not (<code>false</code>).
description	string	Description of a custom picklist value. This field is only relevant for the standard <code>Stage</code> field in opportunities. It is useful to include a description for a customized picklist value so that the historical reason for creating it can be tracked. This field is available in API version 16.0 and later.
forecastCategory	ForecastCategories (enumeration of type string)	<p>Indicates whether this value is associated with a forecast category (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Stage</code> field in opportunities. For more information about forecast categories, including the valid string values listed below, see “Working with Forecast Categories” in the Database.com online help.</p> <ul style="list-style-type: none"> • Omitted • Pipeline • BestCase • Forecast • Closed <p>This field is available in API version 16.0 and later.</p>
fullName	string	The name used as a unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
highPriority	boolean	Indicates whether this value is a high priority item (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Priority</code> field in tasks. For more information about tasks, see “Creating Tasks” in the Database.com online help. This field is available in API version 16.0 and later.

Field Name	Field Type	Description
probability	int	Indicates whether this value is a probability percentage (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Stage</code> field in opportunities. For more information about opportunities, see “Opportunities Overview” in the Database.com online help. This field is available in API version 16.0 and later.
reverseRole	string	A picklist value corresponding to a reverse role name for a partner. If the role is “subcontractor”, then the reverse role might be “general contractor”. Assigning a partner role to an account in Database.com creates a reverse partner relationship so that both accounts list the other as a partner. This field is only relevant for partner roles. For more information, see “Partner Fields” in the Database.com online help. This field is available in API version 18.0 and later.
reviewed	boolean	Indicates whether this value is associated with a reviewed status (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Status</code> field in solutions. For more information about opportunities, see “Creating Solutions” in the Database.com online help. This field is available in API version 16.0 and later.
won	boolean	Indicates whether this value is associated with a closed or won status (<code>true</code>), or not (<code>false</code>). This field is only relevant for the standard <code>Stage</code> field in opportunities. This field is available in API version 16.0 and later.

Java Sample

The following sample uses a picklist. For a complete sample of using a picklist with record types and profiles, see [Profile](#) on page 130.

```
public void setPicklistValues() {
    // Create a picklist
    Picklist expenseStatus = new Picklist();
    PicklistValue unsubmitted = new PicklistValue();
    unsubmitted.setFullName("Unsubmitted");
    PicklistValue submitted = new PicklistValue();
    submitted.setFullName("Submitted");
    PicklistValue approved = new PicklistValue();
    approved.setFullName("Approved");
    PicklistValue rejected = new PicklistValue();
    rejected.setFullName("Rejected");
    expenseStatus.setPicklistValues(new PicklistValue[]
        {unsubmitted, submitted, approved, rejected});

    CustomField expenseStatusField = new CustomField();
    expenseStatusField.setFullName(
        "ExpenseReport__c.ExpenseStatus__c");
    expenseStatusField.setLabel("Expense Report Status");
    expenseStatusField.setType(FieldType.Picklist);
    expenseStatusField.setPicklist(expenseStatus);
    try {
        AsyncResult[] ars =
            metadataConnection.create(new Metadata[] {expenseStatusField});
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Declarative Metadata Sample Definition

The following sample shows usage for picklists, including dependent picklists, in a custom object. The `isAmerican__c` checkbox controls the list of manufacturers shown in the `manufacturer__c` picklist. The `manufacturer__c` checkbox in turn controls the list of models shown in the `model__c` picklist.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  <deploymentStatus>Deployed</deploymentStatus>
  <enableActivities>true</enableActivities>
  <fields>
    <fullName>isAmerican__c</fullName>
    <defaultValue>>false</defaultValue>
    <label>American Only</label>
    <type>Checkbox</type>
  </fields>
  <fields>
    <fullName>manufacturer__c</fullName>
    <label>Manufacturer</label>
    <picklist>
      <controllingField>isAmerican__c</controllingField>
      <picklistValues>
        <fullName>Chrysler</fullName>
        <controllingFieldValues>checked</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
      <picklistValues>
        <fullName>Ford</fullName>
        <controllingFieldValues>checked</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
      <picklistValues>
        <fullName>Honda</fullName>
        <controllingFieldValues>unchecked</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
      <picklistValues>
        <fullName>Toyota</fullName>
        <controllingFieldValues>unchecked</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
      <sorted>>false</sorted>
    </picklist>
    <type>Picklist</type>
  </fields>
  <fields>
    <fullName>model__c</fullName>
    <label>Model</label>
    <picklist>
      <controllingField>manufacturer__c</controllingField>
      <picklistValues>
        <fullName>Mustang</fullName>
        <controllingFieldValues>Ford</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
      <picklistValues>
        <fullName>Taurus</fullName>
        <controllingFieldValues>Ford</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
      <picklistValues>
        <fullName>PT Cruiser</fullName>
        <controllingFieldValues>Chrysler</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
      <picklistValues>
        <fullName>Pacifica</fullName>
        <controllingFieldValues>Chrysler</controllingFieldValues>
        <default>>false</default>
      </picklistValues>
    </picklist>
  </fields>
</CustomObject>
```

```

    </picklistValues>
  </picklistValues>
    <fullName>Accord</fullName>
    <controllingFieldValues>Honda</controllingFieldValues>
    <default>>false</default>
  </picklistValues>
</picklistValues>
  <fullName>Civic</fullName>
  <controllingFieldValues>Honda</controllingFieldValues>
  <default>>false</default>
</picklistValues>
</picklistValues>
  <fullName>Prius</fullName>
  <controllingFieldValues>Toyota</controllingFieldValues>
  <default>>false</default>
</picklistValues>
</picklistValues>
  <fullName>Camry</fullName>
  <controllingFieldValues>Toyota</controllingFieldValues>
  <default>>false</default>
</picklistValues>
  <sorted>>false</sorted>
</picklist>
<type>Picklist</type>
</fields>
....
</CustomObject>

```

The following sample shows usage for the standard Stage field in opportunities.

```

<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  <fields>
    <fullName>StageName</fullName>
    <picklist>
      <picklistValues>
        <fullName>Prospecting</fullName>
        <default>>false</default>
        <forecastCategory>Pipeline</forecastCategory>
        <probability>10</probability>
      </picklistValues>
      <picklistValues>
        <fullName>Qualification</fullName>
        <default>>false</default>
        <forecastCategory>Pipeline</forecastCategory>
        <probability>10</probability>
      </picklistValues>
      <picklistValues>
        <fullName>Needs Analysis</fullName>
        <default>>false</default>
        <forecastCategory>Pipeline</forecastCategory>
        <probability>20</probability>
      </picklistValues>
      ...
    </picklist>
  </fields>
</CustomObject>

```

SharingReason

Represents an Apex sharing reason, which is used to indicate why sharing was implemented for a custom object. Apex managed sharing allows developers to use Apex to programmatically share custom objects. When you use Apex managed sharing to share a custom object, only users with the “Modify All Data” permission can add or change the sharing on the custom object's

record, and the sharing access is maintained across record owner changes. For more information, see “Overview of Sharing Settings” in the Database.com online help.

Use `SharingReason` to create, update, or delete sharing reason definitions for a custom object. It extends the `Metadata` metadata type and inherits its `fullName` field.

Version

Sharing reasons are available in API version 14.0 and later.

Fields

Field	Field Type	Description
<code>fullName</code>	string	Required. Sharing reason name. The <code>__c</code> suffix is appended to custom sharing reasons. Inherited from <code>Metadata</code> , this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See <code>create()</code> to see an example of this field specified for a call.
<code>label</code>	string	Required. Descriptive label for the sharing reason. Maximum of 40 characters.

Declarative Metadata Sample Definition

The definition of a sharing reason in a custom object:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  . . .
  <sharingReasons>
    <fullName>recruiter__c</fullName>
    <label>Recruiter</label>
  </sharingReasons>
  . . .
</CustomObject>
```

SharingRecalculation

Represents Apex classes that recalculate the Apex managed sharing for a specific custom object. For more information, see “Recalculating Apex Managed Sharing” in the Database.com online help.

Version

Sharing recalculations are available in API version 14.0 and later.

Fields

Field	Field Type	Description
<code>className</code>	string	Required. The Apex class that recalculates the Apex sharing for a custom object. This class must implement the <code>Database.Batchable</code> interface.

Declarative Metadata Sample Definition

The definition of a sharing recalculation in a custom object:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  . . .
  <sharingRecalculations>
    <className>RecruiterRecalculation</className>
  </sharingRecalculations>
  . . .
</CustomObject>
```

ValidationRule

Represents a validation rule, which is used to verify that the data a user enters in a record is valid and can be saved. A validation rule contains a formula or expression that evaluates the data in one or more fields and returns a value of `true` or `false`.

Validation rules also include an error message that your client application can display to the user when the rule returns a value of `true` due to invalid data. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

As of API version 20.0, validation rules can't have compound fields. Examples of compound fields include addresses, first and last names, dependent picklists, and dependent lookups.

Version

Validation rules are available in API version 12.0 and later.

Fields

Field Name	Field Type	Description
<code>active</code>	boolean	Required. Indicates whether this validation rule is active, (<code>true</code>), or not active (<code>false</code>).
<code>description</code>	string	A description of the validation rule.
<code>errorConditionFormula</code>	string	Required. The validation formula, as documented in the validation formula page. See “Defining Validation Rules” in the Database.com online help.
<code>errorDisplayField</code>	string	The fully specified name of a field in the application. If a value is supplied in this field, the value in <code>errorMessage</code> appears next to the specified field. If you do not specify a value, the error message appears at the top of the page.
<code>errorMessage</code>	string	Required. The message that appears if the validation rule fails. The message must be 255 characters or less.
<code>fullName</code>	string	The internal name of the validation rule, with white spaces and special characters escaped for validity. The name can only contain characters, letters, and the underscore (<code>_</code>) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See <code>create()</code> to see an example of this field specified for a call.

Declarative Metadata Sample Definition

A sample XML definition of a validation rule in a custom object is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
  <deploymentStatus>Deployed</deploymentStatus>
  <fields>
    <fullName>Mommy_Cat__c</fullName>
    <label>Mommy Cat</label>
    <referenceTo>Cat__c</referenceTo>
    <relationshipName>Cats</relationshipName>
    <type>Lookup</type>
  </fields>
  <label>Cat</label>
  <nameField>
    <label>Cat Name</label>
    <type>Text</type>
  </nameField>
  <pluralLabel>Cats</pluralLabel>
  <sharingModel>ReadWrite</sharingModel>
  <validationRules>
    <fullName>CatsRule</fullName>
    <active>true</active>
    <errorConditionFormula>OR(Name = &apos;Milo&apos;,Name =
&apos;Moop&apos;)</errorConditionFormula>
    <validationMessage>Name must be that of one of my cats</validationMessage>
  </validationRules>
</CustomObject>
```

Weblink

Represents a Weblink defined in a custom object. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

Version

Weblinks are available in API version 12.0 and later.

Fields

The Weblink definition contains the following fields.

Field Name	Field Type	Description
availability	WebLinkAvailability (enumeration of type string)	Required. Indicates whether the Weblink is only available online (online, or if it is also available offline (offline).
description	string	A description of the Weblink.
displayType	WebLinkDisplayType (enumeration of type string)	Represents how this Weblink is rendered. Valid values: <ul style="list-style-type: none"> link for a hyperlink button for a button massAction for a button attached to a related list
encodingKey	Encoding (enumeration of type string)	Required. The default encoding setting is Unicode: UTF-8. Change the default encoding setting if the target of a link requires data in a different format. This is available if your Content Source is URL. Valid values include:

Field Name	Field Type	Description
		<ul style="list-style-type: none"> UTF-8: “Unicode (UTF-8)” in the UI ISO-8859-1: “General US & Western Europe (ISO-8859-1, ISO-LATIN-1)” in the UI Shift_JIS: “Japanese (Shift-JIS)” in the UI ISO-2022-JP: “Japanese (JIS)” in the UI EUC-JP: “Japanese (EUC)” in the UI ks_c_5601-1987: “Korean (ks_c_5601-1987)” in the UI Big5: “Traditional Chinese (Big5)” in the UI GB2312: “Simplified Chinese (GB2312)” in the UI BIG5-HKSCS: “Traditional Chinese Hong Kong (Big5-HKSCS)” in the UI x-SJIS_0213: “Japanese (Shift-JIS_2004)” in the UI
fullName	string	<p>The name of the weblink with white spaces and special characters escaped for validity. The name can only contain characters, letters, and the underscore (<code>_</code>) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters.</p> <p>Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See <code>create()</code> to see an example of this field specified for a call.</p>
hasMenuBar	boolean	If the <code>openType</code> is <code>newWindow</code> , whether to show the browser menu bar for the window (<code>true</code> or not (<code>false</code>)). Otherwise this field should not be specified.
hasScrollbars	boolean	If the <code>openType</code> is <code>newWindow</code> , whether to show the scroll bars for the window (<code>true</code>) or not (<code>false</code>). Otherwise this field should not be specified.
hasToolBar	boolean	If the <code>openType</code> is <code>newWindow</code> , whether to show the browser toolbar for the window (<code>true</code>) or not (<code>false</code>). Otherwise this field should not be specified.
height	int	Height in pixels of the window opened by this Weblink. Required if the <code>openType</code> is <code>newWindow</code> , otherwise should not be specified
isResizable	boolean	If the <code>openType</code> is <code>newWindow</code> , whether to allow resizing of the window (<code>true</code>) or not (<code>false</code>). Otherwise this field should not be specified.
linkType	WebLinkType (enumeration of type string)	<p>Required. Represents whether the content of this Weblink is specified by a URL or a JavaScript code block.</p> <p>Valid values:</p> <ul style="list-style-type: none"> url javascript
masterLabel	string	The master label for the Weblink.

Field Name	Field Type	Description
openType	WebLinkWindowType (enumeration of type string)	Required. When this button is clicked, specifies the window style that will be used to display the content. Valid values: <ul style="list-style-type: none"> newWindow sidebar noSidebar replace onClickJavaScript
position	WebLinkPosition (enumeration of type string)	If the <code>openType</code> is <code>newWindow</code> , how the new window should be displayed. Otherwise this field should not be specified. Valid values: <ul style="list-style-type: none"> fullScreen none topLeft
protected	boolean	Required. Indicates whether this sub-component is protected (<code>true</code>) or not (<code>false</code>). Protected sub-components cannot be linked to or referenced by components or sub-components created in the installing organization.
requireRowSelection	boolean	If the <code>openType</code> is <code>massAction</code> , whether to require individual row selection to execute the action for this button (<code>true</code>) or not (<code>false</code>). Otherwise this field should not be specified.
showsLocation	boolean	If the <code>openType</code> is <code>newWindow</code> , whether or not to show the browser location bar for the window; otherwise this field should not be specified.
showsStatus	boolean	If the <code>openType</code> is <code>newWindow</code> , whether or not to show the browser status bar for the window. Otherwise, this field should not be specified.
url	string	If the value of <code>linkType</code> is <code>url</code> , this is the URL value. If the value of <code>linkType</code> is <code>javascript</code> , this is the JavaScript content. If the value neither of these, the this field should not be specified. Content must be escaped in a manner consistent with XML parsing rules.
width	int	Width in pixels of the window opened by this Weblink. Required if the <code>openType</code> is <code>newWindow</code> , otherwise should not be specified.

Java Sample

The following Java sample shows sample values for Weblink fields:

```
public void weblinkSample(String name) throws Exception {
    Weblink weblink = new Weblink();
    // name variable represents the full name of the object
    // on which to create the weblink, for example, customObject__c
    weblink.setFullName(name + ".googleButton");
}
```



```

weblink.setUrl("http://www.google.com");
weblink.setAvailability(WebLinkAvailability.online);
weblink.setLinkType(WebLinkType.url);
weblink.setEncodingKey(Encoding.fromString("UTF-8"));
weblink.setOpenType(WebLinkWindowType.newWindow);
weblink.setHeight(600);
weblink.setWidth(600);
weblink.setShowsLocation(false);
weblink.setHasScrollbars(true);
weblink.setHasToolbar(false);
weblink.setHasMenubar(false);
weblink.setShowsStatus(false);
weblink.setIsResizable(true);
weblink.setPosition(WebLinkPosition.none);
weblink.setMasterLabel("google");
weblink.setDisplayType(WebLinkDisplayType.link);

AsyncResult[] asyncResults = metadataConnection.create(new WebLink[]{weblink});
// After the create() call completes, we must poll the results of checkStatus()
//
}

```

Declarative Metadata Sample Definition

The following is the definition of a Weblink in a custom object.

```

<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
  <webLinks>
    <fullName>googleButton</fullName>
    <availability>online</availability>
    <displayType>link</displayType>
    <encodingKey>UTF-8</encodingKey>
    <hasMenubar>false</hasMenubar>
    <hasScrollbars>true</hasScrollbars>
    <hasToolbar>false</hasToolbar>
    <height>600</height>
    <isResizable>true</isResizable>
    <linkType>url</linkType>
    <masterLabel>google</masterLabel>
    <openType>newWindow</openType>
    <position>none</position>
    <protected>false</protected>
    <showsLocation>false</showsLocation>
    <showsStatus>false</showsStatus>
    <url>http://www.google.com</url>
    <width>600</width>
  </webLinks>
....
</CustomObject>

```

Metadata Field Types

These field types extend the field types described in the *SOAP API Developer's Guide*.

Field Type	Objects	What the Field Contains
CustomField	Custom object Custom field	Represents a custom field.

Field Type	Objects	What the Field Contains
DeleteConstraint	Custom field	A string that represents deletion options for lookup relationships. Valid values are: <ul style="list-style-type: none"> • SetNull • Restrict • Cascade
DeploymentStatus	Custom object Custom field	A string which represents the deployment status of a custom object or field. Valid values are: <ul style="list-style-type: none"> • InDevelopment • Deployed
FieldType	Custom field	Indicates the type of a custom field. Valid values are: <ul style="list-style-type: none"> • AutoNumber • Lookup • MasterDetail • Checkbox • Currency • Date • DateTime • Email • EncryptedText • Number¹ • Percent • Phone • Picklist • MultiselectPicklist • Summary • Text • TextArea • LongTextArea • Summary • Url • Hierarchy • File • CustomDataType • Html • Geolocation <p>¹ A Number custom field is internally represented as a field of type double. Setting the scale of the Number field to 0 gives you a double that behaves like an int.</p>
Gender	Custom object	Indicates the gender of the noun. This is used for languages where words need different treatment depending on their gender. Valid values are: <ul style="list-style-type: none"> • Masculine • Feminine • Neuter

Field Type	Objects	What the Field Contains
		<ul style="list-style-type: none"> Euter (Swedish) Common (Dutch)
Picklist (Including Dependent Picklist)	Custom field	Represents a picklist, a set of labels and values that can be selected from a picklist.
SharingModel	Custom object Custom field	Represents the sharing model for the custom object or custom field. Valid values are: <ul style="list-style-type: none"> Private Read ReadWrite
StartsWith	Custom object Custom field	Indicates whether the noun starts with a vowel, consonant, or is a special character. This is used for languages where words need different treatment depending on the first character. Valid values are: <ul style="list-style-type: none"> Consonant Vowel Special (for nouns starting with z, or s plus consonants)
TreatBlanksAs	Custom field	Indicates how blanks should be treated. Valid values are: <ul style="list-style-type: none"> BlankAsBlank BlankAsZero

ExternalDataSource



Note: Sunlight Search, which lets you access external data sources, including SharePoint data in Salesforce via external objects, is currently available as a beta feature. You can provide feedback and suggestions on the [IdeaExchange](#). For information on enabling this feature for your organization, contact salesforce.com.

Represents the metadata associated with an external data source. Create external data sources to manage connection details for integration with data sources outside of Salesforce. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ExternalDataSource components are stored in the `dataSources` directory of the corresponding package directory. ExternalDataSource components have the extension `.dataSource`, and the suffix is the name of the external data source.

Version

ExternalDataSource components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
<code>apiKey</code>	string	The key to connect to the external data source.

Field Name	Field Type	Description
certificate	string	Available only for SharePoint 2010 and 2007. If a security certificate is required, click the Certificate Lookup icon to search across active certificates created in Salesforce.
customConfiguration	string	A custom string used for configuring the data source.
endpoint	string	The URL which connects to the data source.
label	string	A user-friendly name for the data source displayed in the Salesforce user interface, such as in list views. Examples include Acme Team Marketing Site, or Acme SharePoint.
oauthRefreshToken	string	The refreshed OAuth token. Used to obtain new access tokens for an existing end user during a specified period.
oauthScope	string	The scope of the access token as a list of space-delimited strings for permission settings that specifies access to a specific user's data.
oauthToken	string	The access token issued by your authorization server.
password	string	The password to be used by Salesforce to access the external source. Ensure that the credentials you use have adequate privileges to be able to access the data source, perform searches, and return information about the external data. Depending on how you set up access, you may need to provide the administrator password.
principalType	ExternalPrincipal Type (enumeration of type string)	The identity type used to authenticate to the data source outside Salesforce. On the External Data Source Setup and Edit pages, these selections are made from the Identity Type field. The valid values are: <ul style="list-style-type: none"> Anonymous PerUser NamedUser
protocol	Authentication Protocol (enumeration of type string)	The protocol required to access the data source outside Salesforce. If your External Data Source is SharePoint, select Basic Authentication . The valid values are: <ul style="list-style-type: none"> NoAuthentication OAuth Password
remoteIdentifier	string	Available beginning in API version 29.0. The server URL or default external repository.
repository	string	Used for SharePoint Online. If metadata is not accessible, use this field to create tables and default table fields.
type	string	The type of data source you want to connect to. The valid values are: <ul style="list-style-type: none"> OData Sharepoint SharepointOnline
username	string	The username to be used by Salesforce to access the external source. Ensure that the credentials you use have adequate privileges to be

Field Name	Field Type	Description
		able to access the data source, perform searches, and return information. Depending on how you set up access, you may need to provide the administrator username.
version	string	The version number of the data source you're using.

Group

Represents a set of public groups, which can have users, roles, and other groups.

Declarative Metadata File Suffix and Directory Location

The file suffix for group components is `.group` and components are stored in the `groups` directory of the corresponding package directory.

Version

Group components are available in API version 24.0 and later.

Fields

This metadata type represents the valid values that define a group:

Field Name	Field Type	Description
doesIncludeBosses	boolean	Indicates whether the managers have access (<code>true</code>) or do not have access (<code>false</code>) to records shared with members of the group. This field is only available for public groups.
fullName	string	The unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component. Corresponds to Group Name in the user interface.
name	string	Required. The name of the group. Corresponds to Label in the user interface.

Declarative Metadata Sample Definition

The following is the definition of a group.

```
<?xml version="1.0" encoding="UTF-8"?>
<Group xmlns="http://soap.sforce.com/2006/04/metadata">
  <doesIncludeBosses>true</doesIncludeBosses>
  <fullName>admin</fullName>
  <name>test</name>
</Group>
```

InstalledPackage

Represents a package to be installed or uninstalled. Deploying a newer version of a currently installed package upgrades the package.



Note: You can't deploy a package along with other metadata types. Hence, `InstalledPackage` must be the only metadata type specified in the manifest file.

File Suffix and Directory Location

The package is specified in the `installedPackages` directory, in a file named after the package's namespace prefix. The file extension is `.installedPackage`.

Version

`InstalledPackage` is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
<code>versionNumber</code>	string	The version number of the package. This has the format <i>majorNumber.minorNumber.patchNumber</i> (for example, 2.1.3).
<code>password</code>	string	An optional field specifying the package password.

Declarative Metadata Sample Definition

This specifies a sample package to be installed or uninstalled.

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

Layout

Represents the metadata associated with a page layout. For more information, see “Managing Page Layouts” in the Database.com online help. It extends the `Metadata` metadata type and inherits its `fullName` field.



Note: If you want to edit the Ideas layout, you must specify it by name in the `package.xml` file. In `package.xml`, use the following code to retrieve the Ideas layout:

```
<types>
  <members>Idea-Idea Layout</members>
  <name>Layout</name>
</types>
```

File Suffix and Directory Location

Layouts are stored in the `layouts` directory of the corresponding package directory. The extension is `.layout`.



Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Version

Layouts are available in API version 13.0 and later.

Fields

This metadata type represents the valid values that define a page layout:

Field Name	Field Type	Description
customButtons	string[]	The custom buttons for this layout. Each button is a reference to a Weblink on the same object. For example, a ButtonLink refers to a Weblink on the same standard or custom object named 'ButtonLink'.
customConsoleComponents	CustomConsoleComponents	Represents custom console components (Visualforce pages or lookup fields; Force.com Canvas apps not available) on a page layout. Custom console components only display in the Salesforce console.
emailDefault	boolean	Only relevant if showEmailCheckbox is set; indicates the default value of that checkbox.
excludeButtons	string[]	List of standard buttons to exclude from this layout. For example, <code><excludeButtons>Delete</excludeButtons></code> , will exclude the Delete button from this layout.
headers	LayoutHeader[] (enumeration of type string)	Layout headers are currently only used for tagging, and only appear in the UI if tagging is enabled. For more information, see “Tags Overview” in the Database.com online help. Valid string values are: <ul style="list-style-type: none"> PersonalTagging—tag is private to user. PublicTagging—tag can be viewed by any other user who can access the record.
layoutSections	LayoutSection []	The main sections of the layout containing fields, s-controls, and custom links. The order here determines the layout order.
miniLayout	MiniLayout	A mini layout is used in the mini view of a record in the Console tab, hover details, and event overlays.
multilineLayoutFields	string[]	Fields for the special multiline layout fields which appear in OpportunityProduct layouts. These are otherwise similar to <code>miniLayoutFields</code> miniLayout .
quickActionList	QuickActionList	The list of actions associated with the page layout. This field is available in API version 28.0 and later.
relatedLists	RelatedListItem []	The related lists for the layout, listed in the order they appear in the user interface.

Field Name	Field Type	Description
relatedObjects	string[]	The list of related objects that appears in the mini view of the console. In database terms, these are foreign key fields on the object for the layout. For more information, see “Choosing Related Objects for the Agent Console’s Mini View” in the Database.com online help.
runAssignmentRulesDefault	boolean	Only relevant if showRunAssignmentRulesCheckbox is set; indicates the default value of that checkbox.
showEmailCheckbox	boolean	Only allowed on Case, CaseClose, and Task layouts. If set, a checkbox appears to show email.
showHighlightsPanel	boolean	If set, the highlights panel displays on pages in the Salesforce console. This field is available in API version 22.0 and later.
showInteractionLogPanel	boolean	If set, the interaction log displays on pages in the Salesforce console. This field is available in API version 22.0 and later.
showKnowledgeComponent	boolean	Only allowed on Case layouts. If set, the Knowledge sidebar displays on cases in the Salesforce console. This field is available in API version 20.0 and later.
showRunAssignmentRulesCheckbox	boolean	Only allowed on Lead and Case objects. If set, a checkbox appears on the page to show assignment rules.
showSolutionSection	boolean	Only allowed on CaseClose layout. If set, the built-in solution information section shows up on the page.
showSubmitAndAttachButton	boolean	Only allowed on Case layout. If set, the Submit & Add Attachment button displays on case edit pages to portal users in the Customer Portal.

CustomConsoleComponents

Represents custom console components (Visualforce pages or lookup fields; Force.com Canvas apps not available) on a page layout. Custom console components only display in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
primaryTabComponents	PrimaryTabComponents	Represents custom console components on primary tabs in the Salesforce console. Available in API version 25.0 and later.
subtabComponents	SubtabComponents	Represents custom console components on subtabs in the Salesforce console. Available in API version 25.0 and later.

PrimaryTabComponents

Represents custom console components on primary tabs in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
component	ConsoleComponent []	Represents a custom console component (Visualforce page or lookup field; Force.com Canvas apps not available) on a section of a page

Field Name	Field Type	Description
		layout. Custom console components only display in the Salesforce console. This field is available in API version 29.0 and earlier.
containers	Container[]	Represents a location and style in which to display more than one custom console component on the sidebars of the Salesforce console. You can specify up to five components for each of the four locations (left, right, top, and bottom). This field is available in API version 30.0 and later.

ConsoleComponent

Represents a custom console component (Visualforce page or lookup field; Force.com Canvas apps not available) on a section of a page layout. Custom console components only display in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
height	int	Required for components with a location of top or bottom. The height of the custom console component. The value must be specified in pixels and be greater than 0 but less than 999.
location	string	Required. The location of the custom console component on the page layout. Valid values are right, left, top, and bottom. A component can have one location for each page layout.
visualforcePage	string	Required. The unique name of the custom console component. For example, ConsoleComponentPage.
width	int	Required for components with a location of left or right. The width of the custom console component. The value must be specified in pixels and be greater than 0 but less than 999.

Container

Represents a location and style in which to display more than one custom console component in the sidebars of the Salesforce console. For example, you can display multiple components in the right sidebar of the console with a style of either stack, tabs, or accordion. Available in API version 30.0 and later.

Field Name	Field Type	Description
height	int	Required for components with a location of top or bottom. The height of the components' container. The unit of measurement, in pixels or percent, is determined by the unit field.
region	string	Required. The location of the components' container. Valid values include: <ul style="list-style-type: none"> right left top bottom
sidebarComponent	SidebarComponent[]	Represents a specific custom console component to display in the components' container.
style	string	Required. The style of the container in which to display multiple components. Valid values include: <ul style="list-style-type: none"> stack—a content area with multiple frames.

Field Name	Field Type	Description
		<ul style="list-style-type: none"> tabs—a single content area with a list of multiple panels. accordian—a collapsible content area.
unit	string	<p>Required. The unit of measurement, in pixels or percent, for the height or width of the components' container.</p> <p>Pixel values are simply the number of pixels, for example, 500, and must be greater than 0 but less than 999. Percentage values must include the percent sign, for example, 20%, and must be greater than 0 but less than 100.</p>
width	int	Required for components with a location of right or left. The width of the components' container. The unit of measurement, in pixels or percent, is determined by the unit field.

SidebarComponent

Represents a specific custom console component to display in a container that hosts multiple components in one of the sidebars of the Salesforce console. You can specify up to five components for each of the four container locations (left, right, top, and bottom). Available in API version 30.0 and later.

Field Name	Field Type	Description
height	int	Required for components with a location of top or bottom. The height of the component in the container. The unit of measurement, in pixels or percent, is determined by the unit field.
label	string	The name of the component as it displays to console users. Available for components in a container with the style of tabs or accordian.
lookup	string	If the component is a lookup field, the name of the field.
page	string	If the component is a Visualforce page, the name of the Visualforce page.
unit	string	<p>Required. The unit of measurement, in pixels or percent, for the height or width of the component in the container.</p> <p>Pixel values are simply the number of pixels, for example, 500, and must be greater than 0 but less than 999. Percentage values must include the percent sign, for example, 20%, and must be greater than 0 but less than 100.</p>
width	int	Required for components with a location of right or left. The width of the component in the container. The unit of measurement, in pixels or percent, is determined by the unit field.

SubtabComponents

Represents custom console components on subtabs in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
component	ConsoleComponent []	Represents a custom console component (Visualforce page or lookup field; Force.com Canvas apps not available) on a section of a page layout. Custom console components only display in the Salesforce console. This field is available in API version 29.0 and earlier.

Field Name	Field Type	Description
containers	Container []	Represents a location and style in which to display more than one custom console component on the sidebars of the Salesforce console. You can specify up to five components for each of the four locations (left, right, top, and bottom). This field is available in API version 30.0 and later.

MiniLayout

Represents a mini view of a record in the Console tab, hover details, and event overlays.

Field Name	Field Type	Description
fields	string[]	The fields for the mini-layout, listed in the order they appear in the UI. Fields that appear here must appear in the main layout.
relatedLists	RelatedListItem []	The mini related list, listed in the order they appear in the UI. You cannot set sorting on mini related lists. Fields that appear here must appear in the main layout.

LayoutSection

LayoutSection represents a section of a page layout, such as the Custom Links section.

Field Name	Field Type	Description
customLabel	boolean	Indicates if this section's label is custom or standard (built-in). Custom labels can be any text, but must be translated. Standard labels have a predefined set of valid values, for example 'System Information', which are automatically translated.
detailHeading	boolean	Controls if this section appears in the detail page. In the UI, this setting corresponds to the checkbox in the section details dialog.
editHeading	boolean	Controls if this section appears in the edit page.
label	string	The label; either standard or custom, based on the <code>customLabel</code> flag.
layoutColumns	LayoutColumn []	The columns of the layout, depending on the style. There may be 1, 2, or 3 columns, ordered left to right.
style	LayoutSectionStyle (enumeration of type string)	The style of the layout: <ul style="list-style-type: none"> <code>TwoColumnsTopToBottom</code> - Two columns, tab goes top to bottom <code>TwoColumnsLeftToRight</code> - Two columns, tab goes left to right <code>OneColumn</code> - One column <code>CustomLinks</code> - Contains custom links only
summaryLayout	SummaryLayout	Reserved for future use.

LayoutColumn

LayoutColumn represents the items in a column within a layout section.

Field Name	Field Type	Description
layoutItems	LayoutItem[]	The individual items within a column (ordered from top to bottom).
reserved	string	This field is reserved for Database.com. The field resolves an issue with some SOAP libraries. Any value entered in the field is ignored.

LayoutItem

LayoutItem represents the valid values that define a layout item. An item must have only one of the following set: component, customLink, field, scontrol, page, reportChartComponent.

Field Name	Field Type	Description
behavior	UiBehavior (enumeration of type string)	Determines the field behavior. Valid string values: <ul style="list-style-type: none"> Edit - The layout field can be edited but is not required Required - The layout field can be edited and is required Readonly - The layout field is read-only
component	string	Reference to a component. Value must be <code>sfa:socialCard</code> . This field is available in API version 30.0 and later. This is only allowed inside a <code>RelatedContentItem</code> . <code>sfa:socialCard</code> is only supported on page layouts for contacts, accounts, and leads.
customLink	string	The <code>customLink</code> reference. This is only allowed inside a <code>CustomLink</code> layoutSection.
emptySpace	boolean	Controls if this layout item is a blank space.
field	string	The field name reference, relative to the layout object, for example <code>Description</code> or <code>MyField__c</code> .
height	int	For s-control and pages only, the height in pixels.
page	string	Reference to a Visualforce page.
reportChartComponent	ReportChartComponentLayoutItem	Refers to a report chart that you can add to a standard or custom object page.
scontrol	string	Reference to an s-control.
showLabel	boolean	For s-control and pages only, whether or not to show the label.
showScrollbars	boolean	For s-control and pages only, whether or not to show scrollbars.
width	string	For s-control and pages only, the width in pixels or percent. Pixel values are simply the number of pixels, for example, 500. Percentage values must include the percent sign, for example, 20%.

ReportChartComponentLayoutItem

Represents the settings for a report chart on a standard or custom page.

Field Name	Field Type	Description
contextFilterableField	string	Unique development name of the field by which a report chart is filtered to return data relevant to the page. If set, chart data is filtered by the ID field for the parent object of the page or report type. The parent object for the report type and the page must match for a chart to return relevant data.
includeContext	boolean	If <code>true</code> , filters the report chart to return data that's relevant to the page.
reportName	string	Unique development name of a report that includes a chart.
showTitle	boolean	If <code>true</code> , applies the title from the report to the chart.
size	ReportChartComponentSize (enumeration of type string)	The chart size is medium when no value is specified. Valid values: <ul style="list-style-type: none"> • SMALL • MEDIUM • LARGE

QuickActionList

QuickActionList represents the list of actions associated with the page layout. Available in API version 28.0 and later.

Field Name	Field Type	Description
quickActionListItems	QuickActionListItem []	Array of zero or more QuickActionList objects.

QuickActionListItem

QuickActionListItem represents an action in the QuickActionList. Available in API version 28.0 and later.

Field Name	Field Type	Description
quickActionName	string	The API name of the action.

RelatedListItem

RelatedListItem represents a related list in a page layout.

Field Name	Field Type	Description
customButtons	string[]	A list of custom buttons used in the related list. For more information, see “Defining Custom Buttons and Links” in the Database.com online help.
excludeButtons	string[]	A list of excluded related-list buttons.
fields	string[]	A list of fields displayed in the related list. Retrieval of standard fields on related lists uses aliases instead of field or API names. For example, the <code>Fax</code> , <code>Mobile</code> , and <code>Home Phone</code> fields are retrieved as <code>Phone2</code> , <code>Phone3</code> , and <code>Phone4</code> , respectively.

Field Name	Field Type	Description
relatedList	string	Required. The name of the related list.
sortField	string	The name of the field that is used for sorting.
sortOrder	SortOrder (enumeration of type string)	If the <code>sortField</code> is set, the <code>sortOrder</code> field determines the sort order. <ul style="list-style-type: none"> Asc - sort in ascending order Desc - sort in descending order

SummaryLayout

Controls the appearance of the highlights panel, which summarizes key fields in a grid at the top of a page layout, when Case Feed is enabled. Available in API version 25.0 and later.

Field Name	Field Type	Description
masterLabel	string	Required. The name of the layout label.
sizeX	int	Required. Number of columns in the highlights panel. This must be between 1 and 4 (inclusive).
sizeY	int	Required. Number of rows in each column. This must be either 1 or 2.
sizeZ	int	Reserved for future use. If provided, the setting is not visible to users.
summaryLayoutItems	SummaryLayoutItem[]	Controls the appearance of an individual field and its column and row position within the highlights panel grid, when Case Feed is enabled. At least one is required.
summaryLayoutStyle	SummaryLayoutStyle (enumeration of type string)	Highlights panel style. Valid string values are: <ul style="list-style-type: none"> Default QuoteTemplate DefaultQuoteTemplate CaseInteraction QuickActionLayoutLeftRight (Available in API version 28.0 and later.) QuickActionLayoutTopDown (Available in API version 28.0 and later.)

SummaryLayoutItem

Controls the appearance of an individual field and its column and row position within the highlights panel grid, when Case Feed is enabled. You can have two fields per each grid in a highlights panel. Available in API version 25.0 and later.

Field Name	Field Type	Description
customLink	string	If the item is a custom link, this is the <code>customLink</code> reference.
field	string	The field name reference, relative to the page layout. Must be a standard or custom field that also exists on the detail page.
posX	int	Required. The item's column position in the highlights panel grid. Must be within the range of <code>sizeX</code> .

Field Name	Field Type	Description
posY	int	Required. The item's row position in the highlights panel grid. Must be within the range of <code>sizeY</code> .
posZ	int	Reserved for future use. If provided, the setting is not visible to users.

Declarative Metadata Sample Definition

The following is the definition of a page layout:

```
<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
  <customConsoleComponents>
    <primaryTabComponents>
      <container>
        <region>left</region>
        <style>Stack</style>
        <unit>Pixel</unit>
        <width>101</width>
        <sidebarComponent>
          <width>60</width>
          <page>simplepage1</page>
          <unit>Percentage</unit>
        </sidebarComponent>
        <sidebarComponent>
          <width>40</width>
          <page>Hello_World</page>
          <unit>Percentage</unit>
        </sidebarComponent>
      </container>
    </primaryTabComponents>
    <subtabComponents>
      <component>
        <location>top</location>
        <visualforcePage>ConsoleComponentPage2</visualforcePage>
        <height>200</height>
      </component>
    </subtabComponents>
  </customConsoleComponents>
  <customButtons>ButtonLink</customButtons>
  <layoutSections>
    <editHeading>>true</editHeading>
    <label>Information</label>
    <layoutColumns>
      <layoutItems>
        <behavior>Required</behavior>
        <field>Name</field>
      </layoutItems>
      <layoutItems>
        <height>180</height>
        <scontrol>LayoutSControl</scontrol>
        <showLabel>>true</showLabel>
        <showScrollbars>>true</showScrollbars>
        <width>50%</width>
      </layoutItems>
      <layoutItems>
        <reportChartComponent>
          <contextFilterableField>CUST_ID</contextFilterableField>
          <includeContext>>true</includeContext>
          <reportName>Open_Accounts_by_Cases</reportName>
          <showTitle>>false</showTitle>
          <size>LARGE</size>
        </reportChartComponent>
      </layoutItems>
    </layoutColumns>
  </layoutSections>
</Layout>
```

```

<layoutColumns>
  <layoutItems>
    <behavior>Edit</behavior>
    <field>OwnerId</field>
  </layoutItems>
  <layoutItems>
    <behavior>Edit</behavior>
    <field>CurrencyIsoCode</field>
  </layoutItems>
</layoutColumns>
<style>TwoColumnsTopToBottom</style>
</layoutSections>
<layoutSections>
  <editHeading>true</editHeading>
  <label>System Information</label>
  <layoutColumns>
    <layoutItems>
      <behavior>Readonly</behavior>
      <field>CreatedById</field>
    </layoutItems>
    <layoutItems>
      <behavior>Readonly</behavior>
      <field>Alpha1__c</field>
    </layoutItems>
  </layoutColumns>
  <layoutColumns>
    <layoutItems>
      <behavior>Readonly</behavior>
      <field>LastModifiedById</field>
    </layoutItems>
    <layoutItems>
      <behavior>Edit</behavior>
      <field>TextArea__c</field>
    </layoutItems>
  </layoutColumns>
  <style>TwoColumnsTopToBottom</style>
</layoutSections>
<layoutSections>
  <customLabel>true</customLabel>
  <detailHeading>true</detailHeading>
  <label>Custom Links</label>
  <layoutColumns>
    <layoutItems>
      <customLink>CustomWebLink</customLink>
    </layoutItems>
  </layoutColumns>
  <style>CustomLinks</style>
</layoutSections>
<relatedContent>
  <relatedContentItems>
    <layoutItem>
      <component>sfa:socialPanel</component>
    </layoutItem>
  <relatedContentItems>
</relatedContent>
<miniLayoutFields>Name</miniLayoutFields>
<miniLayoutFields>OwnerId</miniLayoutFields>
<miniLayoutFields>CurrencyIsoCode</miniLayoutFields>
<miniLayoutFields>Alpha1__c</miniLayoutFields>
<miniLayoutFields>TextArea__c</miniLayoutFields>
<miniRelatedLists>
  <relatedList>RelatedNoteList</relatedList>
</miniRelatedLists>
<relatedLists>
  <fields>StepStatus</fields>
  <fields>CreatedDate</fields>
  <fields>OriginalActor</fields>
  <fields>Actor</fields>
  <fields>Comments</fields>
  <fields>Actor.Alias</fields>

```



```

        <fields>OriginalActor.Alias</fields>
        <relatedList>RelatedProcessHistoryList</relatedList>
    </relatedLists>
    <relatedLists>
        <relatedList>RelatedNoteList</relatedList>
    </relatedLists>
</Layout>

```

The following is an example of a layout using `<summaryLayout>`:

```

<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
    <layoutSections>
        <editHeading>true</editHeading>
        <label>System Information</label>
        <layoutColumns>
            <layoutItems>
                <behavior>Readonly</behavior>
                <field>CreatedById</field>
            </layoutItems>
            <layoutItems>
                <behavior>Required</behavior>
                <field>Name</field>
            </layoutItems>
        </layoutColumns>
        <layoutColumns>
            <layoutItems>
                <behavior>Readonly</behavior>
                <field>LastModifiedById</field>
            </layoutItems>
        </layoutColumns>
        <style>TwoColumnsTopToBottom</style>
    </layoutSections>
    <summaryLayout>
        <masterLabel>Great Name</masterLabel>
        <sizeX>4</sizeX>
        <sizeY>2</sizeY>
        <summaryLayoutItems>
            <posX>0</posX>
            <posY>0</posY>
            <field>Name</field>
        </summaryLayoutItems>
    </summaryLayout>
</Layout>

```

The following is an example of a feed-based layout:

```

<Layout>
...
    <feedLayout>
        <leftComponents>
            <componentType>customLinks</componentType>
        </leftComponents>
        <rightComponents>
            <componentType>follow</componentType>
        </rightComponents>
        <rightComponents>
            <componentType>followers</componentType>
        </rightComponents>
        <rightComponents>
            <componentType>visualforce</componentType>
            <page>accountCustomWidget</page>
            <height>200</height>
        </rightComponents>
        <hideSidebar>true</hideSidebar>
        <feedFilterPosition>centerDropDown</feedFilterPosition>
        <feedFilters>
            <feedFilerType>allUpdates</feedFilerType>

```

```

        </feedFilters>
    </feedFilters>
    <feedFilerType>feedItemType</feedFilerType>
    <feedItemType>CallLogPost</feedItemType>
    </feedFilters>
    <feedFilters>
    <feedFilerType>feedItemType</feedFilerType>
    <feedItemType>TextPost</feedItemType>
    </feedFilters>
</feedLayout>
...
</Layout>

```

Metadata

This is the base class for all metadata types. You cannot edit this object. A component is an instance of a metadata type.

Metadata is analogous to sObject, which represents all standard objects. Metadata represents all components and fields in Metadata API. Instead of identifying each component with an ID, each custom object or custom field has a unique `fullName`, which must be distinct from standard object names, as it must be when you create custom objects or custom fields in the Database.com user interface.

Version

Metadata components are available in API version 10.0 and later.

Fields

Field Name	Field Type	Description
<code>fullName</code>	string	Required. The name of the component. If a field, the name must specify the parent object, for example <code>Account.FirstName</code> . The <code>__c</code> suffix must be appended to custom object names and custom field names when you are setting the <code>fullName</code> . For example, a custom field in a custom object could have a <code>fullName</code> of <code>MyCustomObject__c.MyCustomField__c</code> .

See Also:

[CustomObject](#)

[CustomField](#)

[MetadataWithContent](#)

MetadataWithContent

This is the base type for all metadata types that contain content, such as documents or email templates. It extends [Metadata](#). You cannot edit this object.

Version

MetadataWithContent components are available in API version 14.0 and later.

Fields

Field Name	Field Type	Description
content	base64Binary	Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client.
fullName	string	<p>Required. The name of the component. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</p> <p>Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.</p>

See Also:

[Metadata](#)

Package

Used to specify metadata components to be retrieved as part of a [retrieve\(\)](#) call, or to define a package of components.

Name	Type	Description
apiAccessLevel	APIAccessLevel (enumeration of type string)	<p>Package components have access via dynamic Apex and the API to standard and custom objects in the organization where they are installed. Administrators who install packages may wish to restrict this access after installation for improved security. The valid values are:</p> <ul style="list-style-type: none"> Unrestricted—Package components have the same API access to standard objects as the user who is logged in when the component sends a request to the API. Restricted—The administrator can select which standard objects the components can access. Further, the components in restricted packages can only access custom objects in the current package if the user's permissions allow access to them. <p>For more information, see “About API and Dynamic Apex Access in Packages” in the Database.com online help.</p>
description	string	A short description of the package.
fullName	string	The package name used as a unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and

Name	Type	Description
		not contain two consecutive underscores. This field is inherited from the Metadata component.
namespacePrefix	string	The namespace of the developer organization where the package was created.
objectPermissions	ProfileObjectPermissions[]	Indicates which objects are accessible to the package, and the kind of access available (create, read, update, delete).
setupWeblink	string	The weblink used to describe package installation.
types	PackageTypeMembers[]	The type of component being retrieved.
version	string	Required. The version of the component type.

PackageTypeMembers

Use to specify the name and type of components to be retrieved in a package.

Name	Type	Description
members	string	One or more named components, or the wildcard character (*) to retrieve all custom metadata components of the type specified in the <code><name></code> element. To retrieve a standard object, specify it by name. For example <code><members>Account</members></code> will retrieve the standard Account object.
name	string	The type of metadata component to be retrieved. For example <code><name>CustomObject</name></code> will retrieve one or more custom objects as specified in the <code><members></code> element.

PermissionSet

Represents a set of permissions that's used to grant additional access to one or more users without changing their profile or reassigning profiles. You can use permission sets to grant access, but not to deny access. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

In API version 29.0 and later, you can retrieve and deploy access settings for the following managed components in profiles and permission sets:

- Apex classes
- Custom field permissions
- Custom object permissions
- External data sources

For more information, see Managed Component Access in [Sample package.xml Manifest Files](#) on page 18.

Declarative Metadata File Suffix and Directory Location

Permission sets are stored in the `permissionsets` directory. The file name matches the permission set API name and the extension is `.permissionset`. For example, a permission set with the name `User_Management_Permis` is stored in `permissionsets/User_Management_Permis.permissionset`.

Version

Permission sets are available in API version 22.0 and later.

Fields

Field	Field Type	Description
<code>classAccesses</code>	PermissionSetApexClassAccess []	Indicates which top-level Apex classes have methods that users assigned to this permission set can execute. Available in API version 23.0 and later.
<code>description</code>	string	The permission set description. Limit: 255 characters.
<code>fieldPermissions</code>	PermissionSetFieldPermissions []	Indicates which fields are accessible to a user assigned to this permission set, and the kind of access available (readable or editable). Available in API version 23.0 and later.
<code>label</code>	string	The permission set label. Limit: 80 characters.
<code>objectPermissions</code>	PermissionSetObjectPermissions []	Indicates the objects that are accessible to a user assigned to this permission set, and the kind of access available (create, read, edit, delete, and so on). Available in API version 23.0 and later.
<code>userLicense</code>	string	The <code>User License</code> for the permission set. A user license entitles a user to different functionality within Database.com and determines which profiles and permission sets are available to the user.
<code>userPermissions</code>	PermissionSetUserPermission []	Specifies an app or system permission (such as “API Enabled”) and whether it’s enabled for this permission set. In API version 28.0 and earlier, this field retrieves all user permissions, enabled or disabled. In API version 29.0 and later, this field retrieves only enabled user permissions.

PermissionSetApexClassAccess

`PermissionSetApexClassAccess` represents the Apex class access for users assigned to a permission set.

Field	Field Type	Description
<code>apexClass</code>	string	Required. The Apex class name.
<code>enabled</code>	boolean	Required. Indicates whether users assigned to this permission set can execute methods in the top-level class (<code>true</code>) or not (<code>false</code>).

PermissionSetFieldPermissions

`PermissionSetFieldPermissions` represents the field permissions for users assigned to a permission set. In API version 30.0 and later, permissions for required fields can’t be retrieved or deployed.

Field	Field Type	Description
editable	boolean	Required. Indicates whether the field can be edited by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>).
field	string	Required. The API name of the field (such as <code>Warehouse__c.Description__c</code>).
readable	boolean	Indicates whether the field can be read by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>).

PermissionSetObjectPermissions

PermissionSetObjectPermissions represents the object permissions for a permission set. Use one of these elements for each permission.

Field	Field Type	Description
allowCreate	boolean	Required. Indicates whether the object referenced by the <code>object</code> field can be created by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>).
allowDelete	boolean	Required. Indicates whether the object referenced by the <code>object</code> field can be deleted by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>).
allowEdit	boolean	Required. Indicates whether the object referenced by the <code>object</code> field can be edited by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>).
allowRead	boolean	Required. Indicates whether the object referenced by the <code>object</code> field can be viewed by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>).
modifyAllRecords	boolean	Required. Indicates whether the object referenced by the <code>object</code> field can be viewed, edited, or deleted by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>), regardless of the sharing settings for the object. This includes private records (records with no parent object). This is similar to the “Modify All Data” user permission, but limited to the individual object level.
object	string	Required. The API name of the object (such as <code>Warehouse__c</code>).
viewAllRecords	boolean	Required. Indicates whether the object referenced by the <code>object</code> field can be viewed by the users assigned to this permission set (<code>true</code>) or not (<code>false</code>), regardless of the sharing settings for the object. This includes private records (records with no parent object). This is similar to the “View All Data” user permission, but limited to the individual object level.

PermissionSetUserPermission

PermissionSetUserPermission represents an app or system permission for a permission set. Use one of these elements for each permission.

Field	Field Type	Description
enabled	boolean	Required. Indicates whether the permission is enabled (<code>true</code>) or disabled (<code>false</code>).
name	string	Required. The name of the permission.

Declarative Metadata Sample Definition

When adding or changing a permission set, you don't need to include all permissions—you only need to include the permissions you're adding or changing.

```
<?xml version="1.0" encoding="UTF-8"?>
<PermissionSet xmlns="http://soap.sforce.com/2006/04/metadata">
  <description>Grants all rights needed for an HR administrator to manage
employees.</description>
  <label>HR Administration</label>
  <userLicense>Salesforce</userLicense>
  <applicationVisibilities>
    <application>JobApps__Recruiting</application>
    <visible>true</visible>
  </applicationVisibilities>
  <userPermissions>
    <enabled>true</enabled>
    <name>APIEnabled</name>
  </userPermissions>
  <objectPermissions>
    <allowCreate>true</allowCreate>
    <allowDelete>true</allowDelete>
    <allowEdit>true</allowEdit>
    <allowRead>true</allowRead>
    <viewAllRecords>true</viewAllRecords>
    <modifyAllRecords>true</modifyAllRecords>
    <object>Job_Request__c</object>
  </objectPermissions>
  <fieldPermissions>
    <editable>true</editable>
    <field>Job_Request__c.Salary__c</field>
    <readable>true</readable>
  </fieldPermissions>
  <pageAccesses>
    <apexPage>Job_Request_Web_Form</apexPage>
    <enabled>true</enabled>
  </pageAccesses>
  <classAccesses>
    <apexClass>Send_Email_Confirmation</apexClass>
    <enabled>true</enabled>
  </classAccesses>
  <tabSettings>
    <tab>Job_Request__c</tab>
    <visibility>Available</visibility>
  </tabSettings>
  <recordTypeVisibilities>
    <recordType>Recruiting.DevManager</recordType>
    <visible>true</visible>
  </recordTypeVisibilities>
</PermissionSet>
```

The following is an example package.xml manifest used to retrieve the PermissionSet metadata for an organization. When you retrieve permission sets, you should also retrieve the related components with assigned permissions. For example, to retrieve `objectPermissions` and `fieldPermissions` for a custom object, you must also retrieve the `CustomObject` component.

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

```

        <name>CustomTab</name>
    </types>
</types>
<types>
    <members>Job_Request__c</members>
    <name>CustomObject</name>
</types>
</types>
<types>
    <members>JobApps__Recruiting</members>
    <name>CustomApplication</name>
</types>
</types>
<types>
    <members>Recruiting.DevManager</members>
    <name>RecordType</name>
</types>
</types>
<types>
    <members>*</members>
    <name>PermissionSet</name>
</types>
<version>30.0</version>
</Package>

```

Profile

Represents a user profile. A profile defines a user's permission to perform different functions within Database.com. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

In API version 29.0 and later, you can retrieve and deploy access settings for the following managed components in profiles and permission sets:

- Apex classes
- Custom field permissions
- Custom object permissions
- External data sources

For more information, see Managed Component Access in [Sample package.xml Manifest Files](#) on page 18.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.profile`. There is one file for each profile, stored in the `profiles` folder in the corresponding package directory.

Version

Profiles are available in API version 10.0 and later.

Fields

The content of a profile returned by Metadata API depends on the content requested in the [RetrieveRequest](#) message. For example, profiles only include field-level security for fields included in custom objects returned in the same [RetrieveRequest](#) as the profiles. The profile definition contains the following fields:

Field Name	Field Type	Description
<code>classAccesses</code>	ProfileApexClassAccess []	Indicates which top-level Apex classes have methods that users assigned to this profile can execute.
<code>custom</code>	boolean	Indicates whether the profile is a custom (<code>true</code>) or standard (<code>false</code>) profile. Available in API version 30.0 and later.

Field Name	Field Type	Description
description	string	The profile description. Limit: 255 characters. Available in API version 30.0 and later.
fieldLevelSecurities	ProfileFieldLevelSecurity []	Indicates which fields are visible to a user assigned to this profile, and the kind of access available (editable or hidden). This field is available in API version 22.0 and earlier.
fieldPermissions	ProfileFieldLevelSecurity []	Indicates which fields are visible to a user assigned to this profile, and the kind of access available (editable or readable). This field is available in API version 23.0 and later.
fullName	string	The name can only contain characters, letters, and the underscore () character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.
loginHours	ProfileLoginHours []	Indicates the hours within which a user with this profile may log in. If not specified, the profile doesn't restrict a user's login hours. This field is available in API version 25.0 and later.
loginIpRanges	ProfileLoginIpRange []	The list of IP address ranges from which users with a particular profile can log in. This field is available in API version 17.0 and later.
objectPermissions	ProfileObjectPermissions []	Indicates which objects are accessible to a user assigned to this profile, and the kind of access available (create, read, edit, delete, and so on). In API version 28.0 and later, this field is only retrieved when <code>allowRead</code> is <code>true</code> .
userLicense	string	The <code>User License</code> for the profile. A user license entitles a user to different functionality within Database.com and determines which profiles and permission sets are available to the user. This field is available in API version 17.0 and later.
userPermissions	ProfileUserPermission []	Specifies a user permission (such as "API Enabled") and whether it's enabled for this profile. This field retrieves only enabled user permissions. Available in API version 29.0 and later.

ProfileApexClassAccess

`ProfileApexClassAccess` determines which top-level Apex classes have methods that users assigned to this profile can execute.

Field Name	Field Type	Description
apexClass	string	Required. The Apex class name.
enabled	boolean	Required. Indicates whether users assigned to this profile can execute methods in the top-level class (<code>true</code>) or not (<code>false</code>).

ProfileFieldLevelSecurity

ProfileFieldLevelSecurity represents the field level security for users assigned to a profile. In API version 30.0 and later, permissions for required fields can't be retrieved or deployed.

Field Name	Field Type	Description
editable	boolean	Required. Indicates whether this field is editable (<code>true</code>) or not (<code>false</code>). In API version 30.0 and later, when deploying a new custom field, this field is <code>false</code> by default.
field	string	Required. Indicates the name of the field.
hidden	boolean	Indicates whether this field is hidden (<code>true</code>) or not (<code>false</code>). This field is available in API version 22.0 and earlier. For portal profiles, this is set to <code>true</code> by default in API version 19.0 and later.
readable	boolean	Indicates whether this field is readable (<code>true</code>) or not (<code>false</code>). This field is available in API version 23.0 and later. It replaces the <code>hidden</code> field. In API version 30.0 and later, when deploying a new custom field, this field is <code>false</code> by default. For portal profiles, this is set to <code>false</code> by default.

ProfileLoginHours

ProfileLoginHours restricts the days and times within which users with a particular profile can log in.

Field Name	Field Type	Description
weekdayStart	string	Specifies the earliest time on that day that a user with this profile may log in. If a start time for a particular day is specified, an end time for that day must be specified as well. Start can't be greater than end for a particular day. <ul style="list-style-type: none"> Valid values for <code>weekday</code>: <code>monday</code>, <code>tuesday</code>, <code>wednesday</code>, <code>thursday</code>, <code>friday</code>, <code>saturday</code>, or <code>sunday</code>. For example, <code>mondayStart</code> indicates the beginning of the login period for Monday. Valid values for <code>Start</code>: the number of minutes since midnight. Must be evenly divisible by 60 (full hours). For example, 300 is 5:00 a.m.

Field Name	Field Type	Description
<code>weekdayEnd</code>	string	<p>Specifies the time on that day by which a user with this profile must log out.</p> <ul style="list-style-type: none"> Valid values for <code>weekday</code>: <code>monday</code>, <code>tuesday</code>, <code>wednesday</code>, <code>thursday</code>, <code>friday</code>, <code>saturday</code>, or <code>sunday</code>. For example, <code>mondayEnd</code> indicates the close of the login period for Monday. Valid values for <code>End</code>: the number of minutes since midnight. Must be evenly divisible by 60 (full hours). For example, <code>1020</code> is 5:00 p.m.

To delete login hour restrictions from a profile that previously had them, you must explicitly include an empty `loginHours` tag without any start or end times.

ProfileLoginIpRange

`ProfileLoginIpRange` IP defines an IP address ranges from which users with a particular profile can log in.

Field Name	Field Type	Description
<code>endAddress</code>	string	Required. The end IP address for the range.
<code>startAddress</code>	string	Required. The start IP address for the range.

ProfileObjectPermissions

`ProfileObjectPermissions` represents a user's access to objects.



Note: In API version 18.0 and later, these permissions are disabled in new custom objects for any profiles in which “View All Data” or “Modify All Data” is disabled.

Field Name	Field Type	Description
<code>allowCreate</code>	boolean	<p>Indicates whether the object referenced by the <code>object</code> field can be created by the users assigned to this profile (<code>true</code>) or not (<code>false</code>).</p> <p>This field is named <code>revokeCreate</code> before version 14.0 and the logic is reversed. The field name change and the update from <code>true</code> to <code>false</code> and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files. The field name change and the update from <code>true</code> to <code>false</code> and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.</p>
<code>allowDelete</code>	boolean	<p>Indicates whether the object referenced by the <code>object</code> field can be deleted by the users assigned to this profile (<code>true</code>) or not (<code>false</code>).</p> <p>This field is named <code>revokeDelete</code> before version 14.0 and the logic is reversed. The field name change and the update from <code>true</code> to <code>false</code> and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.</p>

Field Name	Field Type	Description
allowEdit	boolean	<p>Indicates whether the object referenced by the <code>object</code> field can be edited by the users assigned to this profile (<code>true</code>) or not (<code>false</code>).</p> <p>This field is named <code>revokeEdit</code> before version 14.0 and the logic is reversed. The field name change and the update from <code>true</code> to <code>false</code> and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.</p>
allowRead	boolean	<p>Indicates whether the object referenced by the <code>object</code> field can be seen by the users assigned to this profile (<code>true</code>) or not (<code>false</code>).</p> <p>This field is named <code>revokeRead</code> before version 14.0 and the logic is reversed. The field name change and the update from <code>true</code> to <code>false</code> and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.</p>
modifyAllRecords	boolean	<p>Indicates whether the object referenced by the <code>object</code> field can be read, edited, or deleted by the users assigned to this profile (<code>true</code>) or not (<code>false</code>), regardless of the sharing settings for the object. This is equivalent to the “Modify All Data” user permission limited to the individual object level. This is a new field in API version 15.0.</p> <p> Note: This field is not available for all objects. Refer to the profile in the user interface to determine which objects currently support these permissions. Profiles with "Modify All Data" ignore <code>modifyAllRecords</code> entries in Metadata API and don't return an error if "Modify All Data" is enabled on the profile.</p>
object	string	<p>Required. The name of the object whose permissions are altered by this profile, for example, <code>MyCustomObject__c</code>.</p>
viewAllRecords	boolean	<p>Indicates whether the object referenced by the <code>object</code> field can be read by the users assigned to this profile (<code>true</code>) or not (<code>false</code>), regardless of the sharing settings for the object. This includes private records (records with no parent object). This is equivalent to the “View All Data” user permission limited to the individual object level. This is a new field in API version 15.0.</p> <p> Note: This field is not available for all objects. Refer to the profile in the user interface to determine which objects currently support these permissions. Profiles with "View All Data" ignore <code>viewAllRecords</code> entries in the Metadata API and don't return an error if "View All Data" is enabled on the profile.</p>

ProfileUserPermission

`ProfileUserPermission` represents an app or system permission for a profile. Use one of these elements for each permission.

Field	Field Type	Description
enabled	boolean	Required. Indicates whether the permission is enabled (<code>true</code>) or disabled (<code>false</code>).
name	string	Required. The permission name.

Java Sample

The following sample uses picklists, profiles, record types, and a custom app:

```
public void profileSample() {
    try {
        // Create an expense report record, tab and app...
        CustomObject expenseRecord = new CustomObject();
        expenseRecord.setFullName("ExpenseReport__c");
        expenseRecord.setLabel("Expense Report");
        expenseRecord.setPluralLabel("Expense Reports");

        expenseRecord.setDeploymentStatus(DeploymentStatus.Deployed);
        expenseRecord.setSharingModel(SharingModel.ReadWrite);

        CustomField nameField = new CustomField();
        nameField.setType(FieldType.AutoNumber);
        nameField.setLabel("Expense Report Number");
        nameField.setDisplayFormat("ER-{0000}");
        expenseRecord.setNameField(nameField);

        AsyncResult[] arsExpenseRecord =
            metadataConnection.create(new Metadata[] {expenseRecord});

        Picklist expenseStatus = new Picklist();
        PicklistValue unsubmitted = new PicklistValue();
        unsubmitted.setFullName("Unsubmitted");
        PicklistValue submitted = new PicklistValue();
        submitted.setFullName("Submitted");
        PicklistValue approved = new PicklistValue();
        approved.setFullName("Approved");
        PicklistValue rejected = new PicklistValue();
        rejected.setFullName("Rejected");
        expenseStatus.setPicklistValues(new PicklistValue[] {
            unsubmitted, submitted, approved, rejected
        });

        CustomField expenseStatusField = new CustomField();
        expenseStatusField.setFullName(
            "ExpenseReport__c.ExpenseStatus__c"
        );
        expenseStatusField.setLabel("Expense Report Status");
        expenseStatusField.setType(FieldType.Picklist);
        expenseStatusField.setPicklist(expenseStatus);
        AsyncResult[] arsStatusField =
            metadataConnection.create(new Metadata[]
                {expenseStatusField});

        CustomTab expenseTab = new CustomTab();
        expenseTab.setFullName("ExpenseReport__c");
        expenseTab.setMotif("Custom70: Handsaw");
        expenseTab.setCustomObject(true);
        AsyncResult[] arsTab =
            metadataConnection.create(new Metadata[] {expenseTab});

        CustomApplication application = new CustomApplication();
        application.setFullName("ExpenseForce");
        application.setTab(new String[] {expenseTab.getFullName()});
        AsyncResult[] arsApp =
            metadataConnection.create(new Metadata[] {application});
    }
}
```

```

// Employees and managers have the same app visibility...
ProfileApplicationVisibility appVisibility =
    new ProfileApplicationVisibility();
appVisibility.setApplication("ExpenseForce");
appVisibility.setVisible(true);

Profile employee = new Profile();
employee.setFullName("Employee");
employee.setApplicationVisibilities(
    new ProfileApplicationVisibility[] {appVisibility}
);
AsyncResult[] arsProfileEmp =
    metadataConnection.create(new Metadata[] {employee});

Profile manager = new Profile();
manager.setFullName("Manager");
manager.setApplicationVisibilities(
    new ProfileApplicationVisibility[] {appVisibility}
);
AsyncResult[] arsProfileMgr =
    metadataConnection.create(new Metadata[] {manager});

// But employees and managers have different access
// to the state of the expense sheet
RecordType edit = new RecordType();
edit.setFullName("ExpenseReport__c.Edit");
RecordTypePicklistValue editStatuses =
    new RecordTypePicklistValue();
editStatuses.setPicklist("ExpenseStatus__c");
editStatuses.setValues(new PicklistValue[]
    {unsubmitted, submitted});
edit.setPicklistValues(new RecordTypePicklistValue[]
    {editStatuses});
AsyncResult[] arsRecTypeEdit =
    metadataConnection.create(new Metadata[] {edit});

RecordType approve = new RecordType();
approve.setFullName("ExpenseReport__c.Approve");
RecordTypePicklistValue approveStatuses =
    new RecordTypePicklistValue();
approveStatuses.setPicklist("ExpenseStatus__c");
approveStatuses.setValues(new PicklistValue[]
    {approved, rejected});
approve.setPicklistValues(new RecordTypePicklistValue[]
    {approveStatuses});
AsyncResult[] arsRecTypeApp =
    metadataConnection.create(new Metadata[] {approve});
} catch (ConnectionException ce) {
    ce.printStackTrace();
}
}

```

Usage

When you use the `retrieve()` call to get information about profiles in your organization, the returned `.profile` files only include security settings for the other metadata types referenced in the retrieve request (with the exception of user permissions, IP address ranges, and login hours, which are always retrieved). For example, the `package.xml` file below contains a `types` element that matches all custom objects, so the returned profiles contain object and field permissions for all custom objects in your organization, but do not include permissions for standard objects, such as `Account`, and standard fields.

```

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

```

```

    <name>Profile</name>
  </types>
  <version>30.0</version>
</Package>

```

The wildcard “*” on CustomObject does not match standard objects and this helps to avoid making unintended, high-impact profile changes. If you create a few custom objects in a Developer Edition organization, `retrieve()` the information, and subsequently `deploy()` the custom objects to your production organization, the profile and field-level security for all your standard objects, such as Account, and standard fields are not overwritten unless you explicitly create separate `types` elements for the standard objects or fields.

Metadata API intentionally makes it somewhat difficult to include standard fields in `retrieve()` calls in order to prevent unexpected profile changes. However, you can still retrieve and deploy profile permissions for custom and standard fields in standard objects, such as Account.

The next `package.xml` file allows you to return profile permissions for Account standard and custom fields. Note how the standard Account object is defined in a `types` element by specifying it as a member of a CustomObject type.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>Account</members>
    <name>CustomObject</name>
  </types>
  <types>
    <members>*</members>
    <name>Profile</name>
  </types>
  <version>30.0</version>
</Package>

```

The final `package.xml` file allows you to return profile permissions for the `MyCustomField__c` custom field in the Account object.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>Account.MyCustomField__c</members>
    <name>CustomField</name>
  </types>
  <types>
    <members>*</members>
    <name>Profile</name>
  </types>
  <version>30.0</version>
</Package>

```

Queue

Represents a holding area for items before they are processed.

Declarative Metadata File Suffix and Directory Location

The file suffix for queue components is `.queue` and components are stored in the `queues` directory of the corresponding package directory. This component supports cases, leads, service contracts (if Entitlements are enabled), and custom objects.

Version

Queue components are available in API version 24.0 and later.

Fields

This metadata type represents the valid values that define a queue:

Field Name	Field Type	Description
doesSendEmailToMembers	boolean	Indicates whether emails are sent to queue members (<code>true</code>) or not (<code>false</code>) when a new record is added to the queue.
email	string	The email address of the queue owner.
fullName	string	The unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component. Corresponds to Queue Name in the user interface.
name	string	Required. The name of the queue. Corresponds to Label in the user interface.
queueSubject	QueueSubject[]	Indicates the supported entity types.

QueueSubject

QueueSubject represents an entity type that the queue supports.

Field Name	Field Type	Description
subjectType	string	Valid values are: <ul style="list-style-type: none"> Case Lead ServiceContract Custom objects (e.g. <code>ObjA_c</code>)

Declarative Metadata Sample Definition

The following is the definition of a queue, which supports Case, Lead, and a custom object named ObjA.

```
<?xml version="1.0" encoding="UTF-8"?>
<Queue xmlns="http://soap.sforce.com/2006/04/metadata">
  <doesSendEmailToMembers>true</doesSendEmailToMembers>
  <email>member@company.com</email>
  <fullName>Your Name</fullName>
  <name>memberQueue</name>
  <queueSubject>
    <subjectType>Case</subjectType>
  </queueSubject>
  <queueSubject>
    <subjectType>Lead</subjectType>
  </queueSubject>
  <queueSubject>
    <subjectType>ObjA_c</subjectType>
  </queueSubject>
</Queue>
```


QuickAction

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. The parent objects supported include:

- Account
- Campaign
- Case
- Contact
- Custom objects
- Group
- Lead
- Opportunity



Note: In the application, QuickActions are referred to as actions or publisher actions.

File Suffix and Directory Location

QuickAction components have the suffix `quickAction` and are stored in the `quickActions` folder.

Version

QuickAction components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
<code>canvas</code>	string	If a custom action is created using a canvas app, this identifies the app. Returns the fully-qualified name of the canvas app in the format <code><namespace>__<dev_name></code> , if the quick action type is <code>Canvas</code> ; otherwise, returns null. This field is available in API version 29.0 and later.
<code>description</code>	string	The description of the action.
<code>fieldOverrides</code>	FieldOverride	The specific field that may be overridden within a QuickAction.
<code>height</code>	int	If a custom action is created, this is the height in pixels of the action pane.
<code>icon</code>	string	The icon used to identify the action.
<code>isProtected</code>	boolean	Indicates whether this component is protected (<code>true</code>) or not (<code>false</code>). Protected components cannot be linked to or referenced by components created in the installing organization.
<code>label</code>	string	Identifies the action and displays to users. This is also the default identifier used for the API and managed packages.
<code>page</code>	string	If a custom action is created using a Visualforce page, this identifies the page.

Field Name	Field Type	Description
quickActionLayout	QuickActionLayout	The layout of fields on the action.
standardLabel	QuickActionLabel (enumeration of type string)	Specifies the standard label to use for the action. The valid values are: <ul style="list-style-type: none"> LogACall LogANote New (A new record) NewRecordType (For example, a label with something like “New Idea”) Update UpdateRecordType NewChild (A new child record) NewChildRecordType CreateNew CreateNewRecordType (For example, a label with something like “Create New Idea”) QuickRecordType Quick (A quick record)
targetObject	string	The object for which the action is created and performed. 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. In this case, Contact is the targetObject.
targetParentField	string	The parent object type of the action. Links the target object to the parent object. For example, use Account if the target object is Contact and the parent object is Account.
targetRecordType	string	Specifies which record type to create. Valid values are: <ul style="list-style-type: none"> Business Account Person Account Master
type	QuickActionType (enumeration of type string)	The type of quick action. Valid values are: <ul style="list-style-type: none"> Create VisualforcePage Post LogACall SocialPost Canvas Update
width	int	If a custom action is created, this is the width in pixels of the action pane.

FieldOverride

Represents the field names and their respective formulas and literal values that comprise overrides in a QuickAction.

Field Name	Field Type	Description
field	string	The name of the specific field to allow overrides on.
formula	string	Specifies the formula to use when overriding a field.
literalValue	string	The value of the field without overrides.

QuickActionLayout

The layout of fields on the action. There is no hard limit to the number of fields you can add to an action layout. However, for optimum usability, we recommend a maximum of eight fields. Adding more than 20 fields can severely impact user efficiency.

Field Name	Field Type	Description
layoutSectionStyle	LayoutSectionStyle (enumeration of type string)	The type of layout structure used. The valid values are: <ul style="list-style-type: none"> TwoColumnsTopToBottom TwoColumnsLeftToRight OneColumn CustomLinks
quickActionLayoutColumns	QuickActionLayoutColumn[]	Specifies columns in a QuickActionLayout.

QuickActionLayoutColumn

A column defined for a QuickActionLayout.

Field Name	Field Type	Description
quickActionLayoutItems	QuickActionLayoutItem []	Specifies row items in a QuickActionLayoutColumn.

QuickActionLayoutItem

A row item comprised of fields and defined for a QuickActionLayoutColumn.

Field Name	Field Type	Description
emptySpace	boolean	Controls if this layout item is a blank space (<code>true</code>) or not (<code>false</code>).
field	string	Represents a specific field in QuickActionLayoutItem. There is no hard limit to the number of fields you can add to an action layout. However, for optimum usability, we recommend a maximum of eight fields. Adding more than 20 fields can severely impact user efficiency.
uiBehavior	UiBehavior (enumeration of type string)	Specifies user input behavior for specific fields in QuickActionLayoutItem. The valid values are: <ul style="list-style-type: none"> Edit Required Readonly

Declarative Metadata Sample Definition

The following is an example of a `QuickAction` component:

```
<?xml version="1.0" encoding="UTF-8"?>
<QuickAction xmlns="http://soap.sforce.com/2006/04/metadata">
  <description>testActionDefinitionTypesCreate</description>
  <fieldOverrides>
    <field>DoNotCall</field>
    <formula>TRUE</formula>
  </fieldOverrides>
  <fieldOverrides>
    <field>LeadSource</field>
    <literalValue>Partner</literalValue>
  </fieldOverrides>
  <label>testActionDefinitionTypesCreate</label>
  <quickActionLayout>
    <layoutSectionStyle>TwoColumnsLeftToRight</layoutSectionStyle>
    <quickActionLayoutColumns>
      <quickActionLayoutItems>
        <emptySpace>false</emptySpace>
        <field>HomePhone</field>
        <uiBehavior>Required</uiBehavior>
      </quickActionLayoutItems>
      <quickActionLayoutItems>
        <emptySpace>true</emptySpace>
        <uiBehavior>Edit</uiBehavior>
      </quickActionLayoutItems>
      <quickActionLayoutItems>
        <emptySpace>false</emptySpace>
        <field>Name</field>
        <uiBehavior>Required</uiBehavior>
      </quickActionLayoutItems>
      <quickActionLayoutItems>
        <emptySpace>false</emptySpace>
        <field>AccountId</field>
        <uiBehavior>Edit</uiBehavior>
      </quickActionLayoutItems>
    </quickActionLayoutColumns>
    <quickActionLayoutColumns>
      <quickActionLayoutItems>
        <emptySpace>false</emptySpace>
        <field>Description</field>
        <uiBehavior>Edit</uiBehavior>
      </quickActionLayoutItems>
    </quickActionLayoutColumns>
  </quickActionLayout>
  <targetObject>Contact</targetObject>
  <targetParentField>Account</targetParentField>
  <type>Create</type>
</QuickAction>
```

RemoteSiteSetting

Represents a remote site setting. `RemoteSiteSetting` extends the [Metadata](#) metadata type and inherits its `fullName` field.


Declarative Metadata File Suffix and Directory Location

`RemoteSiteSetting` components are stored in the `remoteSiteSettings` directory of the corresponding package directory. The file name matches the unique name of the remote site setting, and the extension is `.remoteSite`.

Version

`RemoteSiteSetting` components are available in API version 19.0 and later.

Fields

Field	Field Type	Description
description	string	The description explaining what this remote site setting is used for.
disableProtocolSecurity	boolean	<p>Required. Indicates whether code within Database.com can access the remote site regardless of whether the user's connection is over HTTP or HTTPS (<code>true</code>) or not (<code>false</code>). When <code>true</code>, code within Database.com can pass data from an HTTPS session to an HTTP session, and vice versa.</p> <p> Warning: Only set to <code>true</code> if you understand the security implications.</p>
fullName	string	<p>The name can only contain characters, letters, and the underscore (<code>_</code>) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters.</p> <p>Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.</p>
isActive	boolean	Required. Indicates if the remote site setting is active (<code>true</code>) or not (<code>false</code>).
url	string	Required. The URL for the remote site.

Declarative Metadata Sample Definition

A sample XML definition of a remote site setting is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<RemoteSiteSetting xmlns="http://soap.sforce.com/2006/04/metadata">
  <description>Used for Apex callout to mapping web service</description>
  <disableProtocolSecurity>false</disableProtocolSecurity>
  <isActive>true</isActive>
  <url>https://www.maptestsite.net/mapping1</url>
</RemoteSiteSetting>
```

Role

Represents a role in your organization.

Declarative Metadata File Suffix and Directory Location

The file suffix for role components is `.role` and components are stored in the `roles` directory of the corresponding package directory.

Version

Role components are available in API version 24.0 and later.

Fields

Field Name	Field Type	Description
fullName	string	The unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component. Corresponds to Role Name in the user interface.
parentRole	string	The role above this role in the hierarchy.

Declarative Metadata Sample Definition

The following is the definition of a role.

```
<?xml version="1.0" encoding="UTF-8"?>
<Role xmlns="http://soap.sforce.com/2006/04/metadata">
  <caseAccessLevel>Edit</caseAccessLevel>
  <contactAccessLevel>Edit</contactAccessLevel>
  <description>Sample Role</description>
  <mayForecastManagerShare>>false</mayForecastManagerShare>
  <name>R22</name>
  <opportunityAccessLevel>Read</opportunityAccessLevel>
</Role>
```

SamlSsoConfig

Represents a SAML Single Sign-On configuration. It extends the [Metadata](#) metadata type and inherits its `fullName` field. Single sign-on is a process that allows network users to access all authorized network resources without having to log in separately to each resource. Single sign-on allows you to validate usernames and passwords against your corporate user database or other client application rather than having separate user passwords managed by Database.com.

File Suffix and Directory Location

SamlSsoConfig components have the suffix `.samlssconfig` and are stored in the `samlssconfigs` folder.

Version

SamlSsoConfig components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
attributeName	string	The name of the identity provider's application. Get this from your identity provider.
attributeNameIdFormat	string	For SAML 2.0 only and when <code>identityLocation</code> is set to <code>Attribute</code> . Possible values include <code>unspecified</code> , <code>emailAddress</code> or <code>persistent</code> . All legal values can be found in the "Name Identifier Format Identifiers" section of the Assertions and Protocols SAML 2.0 specification .

Field Name	Field Type	Description
decryptionCertificate	string	The name of the certificate to use for decrypting incoming SAML assertions. This certificate is saved in the organization's Certificate and Key Management list. Available in API version 30.0 and later.
errorUrl	string	The URL of the page users should be directed to if there's an error during SAML login. It must be a publicly accessible page. The URL can be absolute or relative.
identityLocation	SamlIdentityLocationType (enumeration of type string)	The location in the assertion where a user should be identified. Valid values are: <ul style="list-style-type: none"> SubjectNameId — The identity is in the <Subject> statement of the assertion. Attribute — The identity is specified in an <AttributeValue>, located in the <Attribute> of the assertion.
identityMapping	SamlIdentityType (enumeration of type string)	The identifier the service provider uses for the user during Just-in-Time user provisioning . Valid values are: <ul style="list-style-type: none"> Username — The user's salesforce.com username. FederationId — The federation ID from the user object; the identifier used by the service provider for the user. UserId — The user ID from the user's Database.com organization.
issuer	string	The identification string for the Identity Provider.
loginUrl	string	For SAML 2.0 only: The URL where Database.com sends a SAML request to start the login sequence.
logoutUrl	string	For SAML 2.0 only: The URL to direct the user to when they click the Logout link. The default is http://www.salesforce.com .
name	string	The unique name used by the API and managed packages. The name must begin with a letter and use only alphanumeric characters and underscores. The name cannot end with an underscore or have two consecutive underscores.
oauthTokenEndpoint	string	For SAML 2.0 only: The ACS URL used with enabling Database.com as an identity provider in the Web single sign-on OAuth assertion flow.
redirectBinding	boolean	If you're using My Domain, chose the binding mechanism your identity provider requests for your SAML messages. Values are: <ul style="list-style-type: none"> HTTP POST — HTTP POST binding sends SAML messages using base64-encoded HTML forms. HTTP Redirect — HTTP Redirect binding sends base64-encoded and URL-encoded SAML messages within URL parameters.
salesforceLoginUrl	string	The URL associated with login for the Web single sign-on flow.
samlEntityId	string	The issuer in SAML requests generated by Database.com, and is also the expected audience of any inbound SAML Responses. If you don't have domains deployed, this value can be https://saml.salesforce.com or

Field Name	Field Type	Description
		<code>https://saml.database.com</code> . If you have domains deployed, Database.com recommends that you use your custom domain name.
<code>samlVersion</code>	SamlType (enumeration of type string)	The SAML version in use. Valid values are: <ul style="list-style-type: none"> <code>SAML1_1</code> — SAML 1.1 <code>SAML2_0</code> — SAML 2.0
<code>userProvisioning</code>	boolean	If <code>true</code> , Just-in-Time user provisioning is enabled, which creates users on the fly the first time they try to log in. Specify <code>FederationID</code> for the <code>identityMapping</code> value to use this feature.
<code>validationCert</code>	string	The certificate used to validate the request. Get this from your identity provider.

Declarative Metadata Sample Definition

The following is an example of a `SamlSsoConfig` component. The validation certificate string has been truncated for readability.

```
<?xml version="1.0" encoding="UTF-8"?>
<SamlSsoConfig xmlns="http://soap.sforce.com/2006/04/metadata">
  <identityLocation>SubjectNameId</identityLocation>
  <identityMapping>FederationId</identityMapping>
  <issuer>https://my-idp.my.salesforce.com</issuer>
  <loginUrl>
    https://my-idp.my.salesforce.com/idp/endpoint/HttpRedirect
  </loginUrl>
  <logoutUrl>https://www.salesforce.com</logoutUrl>
  <name>SomeCompany</name>
  <oauthTokenEndpoint>
    https://login.salesforce.com/services/oauth2/token?so=00DD0000000JxeI
  </oauthTokenEndpoint>
  <redirectBinding>true</redirectBinding>
  <salesforceLoginUrl>
    https://login.salesforce.com?so=00DD0000000JxeI
  </salesforceLoginUrl>
  <samlEntityId>
    https://saml.salesforce.com/customPath
  </samlEntityId>
  <samlVersion>SAML2_0</samlVersion>
  <userProvisioning>false</userProvisioning>
  <validationCert>
    MIIEojCCA4ggAwIBAgIOATtxsoBFAAAAAD4...
  </validationCert>
</SamlSsoConfig>
```

Settings

Represents the organization settings related to a feature. For example, your password policies, session settings and network access controls are all available in the `SecuritySettings` component type. Not all feature settings are available in the Metadata API. See [Unsupported Metadata Types](#) on page 76 for information on which feature settings are not available.

Settings can be accessed using the specific component member or via wildcard. For example, in the package manifest file you would use the following section to access `SecuritySettings`:

```
<types>
  <members>Security</members>
  <name>Settings</name>
</types>
```


The member format when used in the package manifest is the component metadata type name without the “Settings” suffix, so in the preceding example “Security” is used instead of “SecuritySettings”.

File Suffix and Directory Location

Each settings component gets stored in a single file in the `settings` directory of the corresponding package directory. The filename uses the format `Setting feature.settings`. For example, the `SecuritySettings` file would be `Security.settings`. See “File Suffix and Directory Location” information for the individual settings components to determine the exact filename.

Version

Settings is available in API version 27.0 and later. See the version information for the individual setting component to determine which API version the settings component became available.

Declarative Metadata Sample Definition

The following is an example package manifest used to deploy or retrieve only the `MobileSettings` for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>Mobile</members>
    <name>Settings</name>
  </types>
  <version>27.0</version>
</Package>
```

The following is an example package manifest used to deploy or retrieve all the available settings metadata for an organization, using a wildcard:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>*</members>
    <name>Settings</name>
  </types>
  <version>27.0</version>
</Package>
```

See Also:

[AccountSettings](#)
[ActivitiesSettings](#)
[MobileSettings](#)
[SecuritySettings](#)

AccountSettings

Represents an organization’s account settings for account teams, account owner report, and the **View Hierarchy** link. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See [Settings](#) for more details.

File Suffix and Directory Location

AccountSettings values are stored in the `Account.settings` file in the `settings` folder. The `.settings` files are different from other named components, as there is only one settings file for each settings component.

Version

AccountSettings is available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
<code>enableAccountOwnerReport</code>	boolean	Indicates whether Account Owner Report may (<code>true</code>) or may not (<code>false</code>) be run by all users.
<code>enableAccountTeams</code>	boolean	Indicates whether Account Teams are enabled (<code>true</code>) or not (<code>false</code>). <code>enableAccountTeams</code> cannot be set to <code>false</code> via the Metadata API.
<code>showViewHierarchyLink</code>	boolean	Indicates whether the default View Hierarchy link on all business account detail pages is visible (<code>true</code>) or hidden (<code>false</code>).

Declarative Metadata Sample Definition

The following is an example of the `Account.settings` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountSettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <enableAccountOwnerReport>true</enableAccountOwnerReport>
  <enableAccountTeams>true</enableAccountTeams>
  <showViewHierarchyLink>true</showViewHierarchyLink>
</AccountSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Account settings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>Account</members>
    <name>Settings</name>
  </types>
  <version>29.0</version>
</Package>
```

See Also:

[Settings](#)

ActivitiesSettings

Represents an organization's activity settings, and its user interface settings for the calendar. Use the `ActivitiesSettings` component type to control the following activity settings:

- Configure group and recurring tasks, recurring and multiday events, and email tracking

- Relate multiple contacts to tasks and events (shared activities)
- Display custom logos in meeting requests

Also use the `ActivitiesSettings` component type to control user interface settings for the calendar, including hover links and drag-and-drop editing.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See [Settings](#) for more details.

File Suffix and Directory Location

`ActivitiesSettings` values are stored in the `Activities.settings` file in the `settings` directory. The `.settings` files are different from other named components, as there is only one settings file for each settings component.

Version

`ActivitiesSettings` is available in API versions 28.0 and later.

Fields

Settings for all types listed below are controlled on the Activity settings page or the User Interface settings page as noted.

Field Name	Field Type	Description
<code>enableActivityReminders</code>	boolean	Enables popup activity reminders for an organization. Administrators control this field on the Activity settings page.
<code>enableClickCreateEvents</code>	boolean	Lets users create events in day and weekly calendar views by double-clicking a specific time slot and entering the details of the event in an overlay. Hovering over an event displays an overlay where users can view the event details or delete the event without leaving the page. Administrators use a mini page layout to configure the fields shown in the overlays. Does not support recurring events or multi-person events. Administrators control this field on the User Interface settings page.
<code>enableDragAndDropScheduling</code>	boolean	Lets users create events associated with records by dragging a record from a list view onto a calendar view and entering the details of the event in an overlay. Hovering over an event displays an overlay where users can view the event details or delete the event without leaving the page. Administrators use a mini page layout to configure the fields shown in the overlays. Administrators control this field on the User Interface settings page.
<code>enableEmailTracking</code>	boolean	Enables tracking of outbound HTML emails if an organization uses HTML email templates. Administrators control this field on the Activity settings page.
<code>enableGroupTasks</code>	boolean	Lets users assign independent copies of a new task to multiple users. Administrators control this field on the Activity settings page.
<code>enableListViewScheduling</code>	boolean	Extends the functionality of <code>enableDragAndDropScheduling</code> and <code>enableClickCreateEvents</code> to list view calendars. Administrators control this field on the User Interface settings page.

Field Name	Field Type	Description
<code>enableMultidayEvents</code>	boolean	Enables creation of events that end more than 24 hours after they start. Administrators control this field on the Activity settings page.
<code>enableRecurringEvents</code>	boolean	Enables creation of events that repeat at specified intervals. Administrators control this field on the Activity settings page.
<code>enableRecurringTasks</code>	boolean	Enables creation of tasks that repeat at specified intervals. Administrators control this field on the Activity settings page.
<code>enableSidebarCalendarShortcut</code>	boolean	In the sidebar, displays a shortcut link to a user's last-used calendar view. Administrators control this field on the Activity settings page.
<code>meetingRequestsLogo</code>	string	Available when <code>showCustomLogoMeetingRequests</code> is enabled. Uploads a custom logo. An administrator can select only a logo that has been uploaded to certain folders in the Documents tab. Administrators control this field on the Activity settings page.
<code>showCustomLogoMeetingRequests</code>	boolean	Displays a custom logo in meeting request emails and on a meeting's Web page. Invitees see the logo when a user either invites them to an event or requests a meeting. Administrators control this field on the Activity settings page.
<code>showEventDetailsMultiUserCalendar</code>	boolean	Displays event details on-screen rather than in hover text. Administrators control this field on the Activity settings page.
<code>showHomePageHoverLinksForEvents</code>	boolean	In the calendar section of the Home tab: <ul style="list-style-type: none"> When a user hovers over the subject of an event, a hover link displays an overlay with selected event details. (Hover links are always available in other calendar views.) When a user clicks the subject of an event, displays the event detail page. Administrators use a mini page layout to configure the fields shown in the overlay. Administrators control this field on the User Interface settings page.
<code>showMyTasksHoverLinks</code>	boolean	In the My Tasks section of the Home tab and on the calendar day view: <ul style="list-style-type: none"> When a user hovers over the subject of a task, a hover link displays an overlay with selected task details. When a user clicks the subject of a task, displays the task detail page. Administrators use a mini page layout to configure the fields shown in the overlay. Administrators control this field on the User Interface settings page.
<code>showRequestedMeetingsOnHomePage</code>	boolean	In the Calendar on the Home tab, displays the Requested Meetings subtab, listing the meetings a user has requested but not confirmed.

Field Name	Field Type	Description
		Disabling this feature removes the New Meeting Request button from the calendar on the Home tab.
		Administrators control this field on the Activity settings page.

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Activity settings metadata for an organization:

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

Declarative Metadata Sample Definition

The following is an example of an activity settings file:

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

  <allowUsersToRelateMultipleContactsToTasksAndEvents>true</allowUsersToRelateMultipleContactsToTasksAndEvents>

  <enableActivityReminders>true</enableActivityReminders>
  <enableClickCreateEvents>true</enableClickCreateEvents>
  <enableDragAndDropScheduling>true</enableDragAndDropScheduling>
  <enableEmailTracking>true</enableEmailTracking>
  <enableGroupTasks>true</enableGroupTasks>
  <enableListViewScheduling>true</enableListViewScheduling>
  <enableMultidayEvents>true</enableMultidayEvents>
  <enableRecurringEvents>true</enableRecurringEvents>
  <enableRecurringTasks>true</enableRecurringTasks>
  <enableSidebarCalendarShortcut>true</enableSidebarCalendarShortcut>
  <meetingRequestsLogo>Folder02/logo03.png</meetingRequestsLogo>
  <showCustomLogoMeetingRequests>true</showCustomLogoMeetingRequests>
  <showEventDetailsMultiUserCalendar>true</showEventDetailsMultiUserCalendar>
  <showHomePageHoverLinksForEvents>true</showHomePageHoverLinksForEvents>
  <showMyTasksHoverLinks>true</showMyTasksHoverLinks>
  <showRequestedMeetingsOnHomePage>true</showRequestedMeetingsOnHomePage>
</ActivitiesSettings>
```

BusinessHoursSettings

Represents the metadata used to manage settings for business hours and holidays in entitlements, entitlement templates, campaigns, and cases. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See [Settings](#) for more details.

File Suffix and Directory Location

Business hours and holidays settings are stored in a single file named `businessHours.settings` in the `settings` directory. The `.settings` files are different from other named components, as there is only one settings file for each settings component.

Version

BusinessHoursSettings is available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
businessHours	BusinessHoursEntry []	Represents the application of business hours to entitlements, entitlement templates, campaigns, and cases.
holidays	Holidays []	Represents a holiday and its usage in businessHours.

BusinessHoursEntry

Represents the application of business hours to entitlements, entitlement templates, campaigns, and cases.

Field Name	Field Type	Description
timeZoneId	string	The time zone for the time that defines business hours.
name	string	Name of the business hours. This name should be unique.
active	string	Indicates whether the business hours are active.
default	string	Indicates whether the business hours are used as the default business hours.
mondayStartTime	string	Start time for the business hours on Monday. Uses the format HH:mm:ss.SSSZ.
mondayEndTime	string	End time for the business hours on Monday. Uses the format HH:mm:ss.SSSZ. The value 00:00:00.000Z specifies midnight on Monday.
tuesdayStartTime	string	Start time for the business hours on Tuesday. Uses the format HH:mm:ss.SSSZ.
tuesdayEndTime	string	End time for the business hours on Tuesday. Uses the format HH:mm:ss.SSSZ. The value 00:00:00.000Z specifies midnight on Tuesday.
wednesdayStartTime	string	Start time for the business hours on Wednesday. Uses the format HH:mm:ss.SSSZ.
wednesdayEndTime	string	End time for the business hours on Wednesday. Uses the format HH:mm:ss.SSSZ. The value 00:00:00.000Z specifies midnight on Wednesday.
thursdayStartTime	string	Start time for the business hours on Thursday. Uses the format HH:mm:ss.SSSZ.
thursdayEndTime	string	End time for the business hours on Thursday. Uses the format HH:mm:ss.SSSZ. The value 00:00:00.000Z specifies midnight on Thursday.
fridayStartTime	string	Start time for the business hours on Friday. Uses the format HH:mm:ss.SSSZ.

Field Name	Field Type	Description
fridayEndTime	string	End time for the business hours on Friday. Uses the format HH:mm:ss.SSSZ. The value 00:00:00.000Z specifies midnight on Friday.
saturdayStartTime	string	Start time for the business hours on Saturday. Uses the format HH:mm:ss.SSSZ.
saturdayEndTime	string	End time for the business hours on Saturday. Uses the format HH:mm:ss.SSSZ. The value 00:00:00.000Z specifies midnight on Saturday.
sundayStartTime	string	Start time for the business hours on Sunday. Uses the format HH:mm:ss.SSSZ.
sundayEndTime	string	End time for the business hours on Sunday. Uses the format HH:mm:ss.SSSZ. The value 00:00:00.000Z specifies midnight on Sunday.

Holidays

Represents a holiday and its usage in businessHours.

Field Name	Field Type	Description
name	string	Name of the holiday. This name does not have to be unique.
description	string	The description of the holiday.
isRecurring	string	Indicates whether the holiday is recurring.
activityDate	string	The date of the holiday. Use for non-recurring holidays. Uses the format HH:mm:ss.SSSZ.
recurrenceStartDate	string	The date the holiday starts recurring. Uses the format yyyy-mm-dd.
recurrenceEndDate	string	The date the holiday stops recurring. Uses the format yyyy-mm-dd. Optional.
startTime	string	The start time on the date of the holiday. Uses the format HH:mm:ss.SSSZ. startTime and endTime must be both null or both not null. If they are both null, indicates the whole day.
endTime	string	The end time on the date of the holiday. Uses the format HH:mm:ss.SSSZ. startTime and endTime must be both null or both not null. If they are both null, indicates the whole day.
recurrenceType	string	The recurrence type of the holiday. Valid values are: RecursDaily, RecursEveryWeekday, RecursMonthly, RecursMonthlyNth, RecursWeekly, RecursYearly, RecursYealyNth.
recurrenceInterval	string	The interval of weeks, months, or years the holiday recurs.
recurrenceDayOfWeek	string	The day of week the holiday recurs. Valid values: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.
recurrenceDayOfMonth	string	The day of month the holiday recurs. Valid values: integers 1-31.
recurrenceInstance	string	Valid values: First, Second, Third, Fourth, Last. Only used for recurrenceType RecursMonthlyNth and RecursYearlyNth. For example, if the recurrenceInstance value is First, the holiday recurs on the first Monday of the month every 3 months.

Field Name	Field Type	Description
recurrenceMonthOfYear	string	Valid values: January, February, March, April, May, June, July, August, September, October, November, December.
businessHours	string	The name of the business hours setting that applies to this holiday.

Declarative Metadata Sample Definition

The following is an example `businesshours.settings` metadata file:

```
<?xml version="1.0" encoding="UTF-8"?>
<BusinessHoursSettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <businessHours>
    <active>true</active>
    <default>true</default>
    <fridayEndTime>00:00:00.000Z</fridayEndTime>
    <fridayStartTime>00:00:00.000Z</fridayStartTime>
    <mondayEndTime>00:00:00.000Z</mondayEndTime>
    <mondayStartTime>00:00:00.000Z</mondayStartTime>
    <name>Default</name>
    <saturdayEndTime>00:00:00.000Z</saturdayEndTime>
    <saturdayStartTime>00:00:00.000Z</saturdayStartTime>
    <sundayEndTime>00:00:00.000Z</sundayEndTime>
    <sundayStartTime>00:00:00.000Z</sundayStartTime>
    <thursdayEndTime>00:00:00.000Z</thursdayEndTime>
    <thursdayStartTime>00:00:00.000Z</thursdayStartTime>
    <timeZoneId>America/Los_Angeles</timeZoneId>
    <tuesdayEndTime>00:00:00.000Z</tuesdayEndTime>
    <tuesdayStartTime>00:00:00.000Z</tuesdayStartTime>
    <wednesdayEndTime>00:00:00.000Z</wednesdayEndTime>
    <wednesdayStartTime>00:00:00.000Z</wednesdayStartTime>
  </businessHours>
  <businessHours>
    <active>true</active>
    <default>false</default>
    <fridayEndTime>00:00:00.000Z</fridayEndTime>
    <fridayStartTime>00:00:00.000Z</fridayStartTime>
    <mondayEndTime>15:00:00.000Z</mondayEndTime>
    <mondayStartTime>09:00:00.000Z</mondayStartTime>
    <name>bh1</name>
    <saturdayEndTime>00:00:00.000Z</saturdayEndTime>
    <saturdayStartTime>00:00:00.000Z</saturdayStartTime>
    <sundayEndTime>00:00:00.000Z</sundayEndTime>
    <sundayStartTime>00:00:00.000Z</sundayStartTime>
    <thursdayEndTime>17:00:00.000Z</thursdayEndTime>
    <thursdayStartTime>10:50:00.000Z</thursdayStartTime>
    <timeZoneId>America/Los_Angeles</timeZoneId>
    <tuesdayEndTime>13:00:00.000Z</tuesdayEndTime>
    <tuesdayStartTime>09:00:00.000Z</tuesdayStartTime>
    <wednesdayEndTime>15:00:00.000Z</wednesdayEndTime>
    <wednesdayStartTime>09:00:00.000Z</wednesdayStartTime>
  </businessHours>
  <holidays>
    <activityDate>2013-09-02</activityDate>
    <businessHours>Default</businessHours>
    <businessHours>bh1</businessHours>
    <isRecurring>false</isRecurring>
    <name>Labor Day</name>
  </holidays>
  <holidays>
    <businessHours>bh1</businessHours>
    <isRecurring>true</isRecurring>
    <name>Thanksgiving</name>
    <recurrenceDayOfMonth>21</recurrenceDayOfMonth>
    <recurrenceMonthOfYear>November</recurrenceMonthOfYear>
    <recurrenceStartDate>2013-11-21</recurrenceStartDate>
    <recurrenceType>RecursYearly</recurrenceType>
  </holidays>
</BusinessHoursSettings>
```



```
</holidays>
</BusinessHoursSettings>
```

The following is an example package.xml manifest that references the BusinessHoursSettings definitions:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>BusinessHours</members>
    <name>Settings</name>
  </types>
  <version>29.0</version>
</Package>
```

LiveAgentSettings

Represents an organization's Live Agent settings, such as whether or not Live Agent is enabled. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveAgentSettings values are stored in the `LiveAgent.settings` file in the `settings` directory. The `.settings` files are different from other named components, as there is only one settings file for each settings component.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See [Settings](#) for more details.

Version

LiveAgentSettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
<code>enableLiveAgent</code>	boolean	Indicates whether Live Agent is enabled (<code>true</code>) or not (<code>false</code>).

Declarative Metadata Sample Definition

This is a sample Live Agent settings file.

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

MobileSettings

Represents an organization's mobile settings, such as mobile Chatter settings, whether Mobile Lite is enabled for an organization, and so on. For more information, see "Managing Salesforce Classic Devices" and "Chatter Mobile for BlackBerry Overview" in the Database.com online help.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See [Settings](#) for more details.

Declarative Metadata File Suffix and Directory Location

MobileSettings values are stored in a single file named `Mobile.settings` in the `settings` directory. The `.settings` files are different from other named components, as there is only one settings file for each settings component.



Note: MobileSettings is no longer available in API versions 25.0 and 26.0.

Version

Mobile settings are available in API version 27.0 and later.

Fields

Field	Field Type	Description
<code>chatterMobile</code>	ChatterMobileSettings	The settings for Chatter mobile devices.
<code>salesforceMobile</code>	SFDCMobileSettings	The settings for general users on mobile devices.
<code>touchMobile</code> (Deprecated)	TouchMobileSettings	The settings for touch on mobile devices.

ChatterMobileSettings

Represents your organization's Chatter Mobile settings.

Field	Field Type	Description
<code>IPadAuthorized</code>	boolean	Indicates whether iPad devices are enabled for Chatter Mobile (<code>true</code>) or not (<code>false</code>).
<code>IPhoneAuthorized</code>	boolean	Indicates whether iPhone devices are enabled for Chatter Mobile (<code>true</code>) or not (<code>false</code>).
<code>androidAuthorized</code>	boolean	Indicates whether Android devices are enabled for Chatter Mobile (<code>true</code>) or not (<code>false</code>).
<code>blackBerryAuthorized</code>	boolean	Indicates whether Blackberry devices are enabled for Chatter Mobile (<code>true</code>) or not (<code>false</code>).
<code>enableChatterMobile</code>	boolean	Indicates whether Chatter Mobile has been enabled for your organization (<code>true</code>) or not (<code>false</code>).

Note: Setting this to `true` enables you to set all of the other settings. If you change this setting from `true` to `false`, and also try to change any of the other `ChatterMobile` settings, your deploy will fail with an error.

Field	Field Type	Description
enablePushNotifications	boolean	Indicates whether Chatter push notifications have been enabled for your organization (<code>true</code>) or not (<code>false</code>)
sessionTimeout	MobileSessionTimeout (enumeration of type string)	The length of time after which users without activity are prompted to log out or continue working. Valid values are: <ul style="list-style-type: none"> • Never • OneMinute • FiveMinutes • TenMinutes • ThirtyMinutes

SFDCMobileSettings

Represents your organization's general mobile settings.

Field	Field Type	Description
enableUserToDeviceLinking	boolean	Permanently link users to their mobile devices. Set this option to <code>true</code> only if you want to prevent your users from switching devices without administrative intervention..
enableMobileLite	boolean	Indicates whether your organization has Mobile Lite enabled (<code>true</code>) or not (<code>false</code>).

TouchMobileSettings

These fields are deprecated. Salesforce Touch has been upgraded to the Salesforce1 app.

Field	Field Type	Description
enableTouchBrowserIPad	boolean	Indicates whether your organization has the Salesforce Touch mobile browser app enabled (<code>true</code>) or not (<code>false</code>).
enableTouchAppIPad	boolean	Indicates whether your organization has the Salesforce Touch downloadable app enabled (<code>true</code>) or not (<code>false</code>)

Declarative Metadata Sample Definition

This is a sample `mobile.settings` metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<MobileSettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <chatterMobile>
    <IPadAuthorized>true</IPadAuthorized>
    <IPhoneAuthorized>true</IPhoneAuthorized>
    <androidAuthorized>true</androidAuthorized>
    <blackBerryAuthorized>true</blackBerryAuthorized>
    <enableChatterMobile>true</enableChatterMobile>
  </chatterMobile>
</MobileSettings>
```

```

    <enablePushNotifications>true</enablePushNotifications>
    <sessionTimeout>Never</sessionTimeout>
  </chatterMobile>
  <dashboardMobile>
    <enableDashboardIPadApp>true</enableDashboardIPadApp>
  </dashboardMobile>
  <salesforceMobile>
    <enableUserToDeviceLinking>false</enableUserToDeviceLinking>
    <enableMobileLite>false</enableMobileLite>
  </salesforceMobile>
  <touchMobile>
    <enableTouchBrowserIPad>false</enableTouchBrowserIPad>
    <enableTouchAppIPad>true</enableTouchAppIPad>
  </touchMobile>
</MobileSettings>

```

See Also:[Settings](#)

SecuritySettings

Represents an organization's security settings. Security settings define trusted IP ranges for network access, password and login requirements, and session expiration and security settings.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See [Settings](#) for more details.

Declarative Metadata File Suffix and Directory Location

SecuritySettings values are stored in a single file named `Security.settings` in the `settings` directory. The `.settings` files are different from other named components, as there is only one settings file for each settings component.



Note: SecuritySettings is no longer available in API versions 25.0 and 26.0.

Version


Security settings are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
<code>networkAccess</code>	NetworkAccess	The trusted IP address ranges from which users can always log in without requiring computer activation.
<code>passwordPolicies</code>	PasswordPolicies	The requirements for passwords and logins, and assistance with retrieving forgotten passwords.
<code>sessionSettings</code>	SessionSettings	The settings for session expiration and security.

NetworkAccess

Represents your organization's trusted IP address ranges for network access.

Field	Field Type	Description
ipRanges	IpRange[]	<p>The trusted IP address ranges from which users can always log in without requiring computer activation.</p>  <p>Note: In order to add an IP range, you need to deploy all existing IP ranges, as well as the one you want to add. Otherwise, the existing IP ranges are replaced with the ones you deploy. To remove all the IP ranges in an organization, leave the networkAccess field blank (<networkAccess></networkAccess>).</p>

IpRange

Defines a range of trusted IP addresses for network access.

Field	Field Type	Description
end	string	The IP address that defines the high end of a range of trusted addresses.
start	string	The IP address that defines the low end of a range of trusted addresses.

PasswordPolicies

Represents your organization's password and login policies.

Field	Field Type	Description
apiOnlyUserHomePageURL	string	The URL to which users with the "API Only User" permission are redirected instead of the login page.
complexity	Complexity (enumeration of type string)	<p>Required. The requirement for which types of characters must be used in a user's password. Valid values are:</p> <ul style="list-style-type: none"> NoRestriction—allows any password value and is the least secure option. AlphaNumeric—requires at least one alphabetic character and one number. This is the default value. SpecialCharacters—requires at least one alphabetic character, one number, and one of the following characters: ! # \$ % - _ = + < >.
expiration	Expiration (enumeration of type string)	<p>Required. The length of time until all user passwords expire and must be changed. Valid values are:</p> <ul style="list-style-type: none"> Never ThirtyDays SixtyDays NinetyDays. This is the default value. SixMonths OneYear

Field	Field Type	Description
passwordAssistanceURL	string	The URL that users can click to retrieve forgotten passwords.
passwordAssistanceMessage	string	The text that appears in the Account Lockout email and at the bottom of the Confirm Identity screen for users resetting their passwords.
historyRestriction	string	Required. The number of previous passwords saved for users so that they must always reset a new, unique password. Valid values are 0 through 15 passwords remembered. The default value is 3.
lockoutInterval	LockoutInterval (enumeration of type string)	Required. The duration of the login lockout. Valid values are: <ul style="list-style-type: none"> FifteenMinutes. This is the default value. ThirtyMinutes SixtyMinutes Forever (must be reset by admin)
maxLoginAttempts	MaxLoginAttempts (enumeration of type string)	Required. The number of login failures allowed for a user before they become locked out. Valid values are: <ul style="list-style-type: none"> NoLimit ThreeAttempts FiveAttempts TenAttempts. This is the default value.
minPasswordLength	MinPasswordLength (enumeration of type string)	Required. The minimum number of characters required for a password. Valid values are: <ul style="list-style-type: none"> FiveCharacters EightCharacters. This is the default value. TenCharacters
obscureSecretAnswer	boolean	Hides the secret answer associated with a password (true) or not (false). <p> Note: If your organization uses the Microsoft Input Method Editor (IME) with the input mode set to Hiragana, when you type ASCII characters they're converted into Japanese characters in normal text fields. However, the IME does not work properly in fields with obscured text. If your organization's users cannot properly enter their passwords or other values after enabling this feature, disable the feature.</p>

Field	Field Type	Description
questionRestriction	QuestionRestriction (enumeration of type string)	Required. The restriction on whether the answer to the password hint question can contain the password itself. Valid values are: <ul style="list-style-type: none"> None DoesNotContainPassword. This is the default value.

SessionSettings

Represents your organization's session expiration and security settings.

Field	Field Type	Description
disableTimeoutWarning	boolean	Indicates whether the session timeout warning popup is disabled (<code>true</code>) or enabled (<code>false</code>).
enableCSRFOnGet	boolean	Indicates whether Cross-Site Request Forgery (CSRF) protection on GET requests on non-setup pages is enabled (<code>true</code>) or disabled (<code>false</code>).
enableCSRFOnPost	boolean	Indicates whether Cross-Site Request Forgery (CSRF) protection on POST requests on non-setup pages is enabled (<code>true</code>) or disabled (<code>false</code>).
enableCacheAndAutocomplete	boolean	Indicates whether the user's browser is allowed to store user names and auto-fill the <code>User Name</code> field on the login page (<code>true</code>) or not (<code>false</code>).
enableClickjackNonsetupSFDC	boolean	Indicates whether clickjack protection for non-setup Database.com pages is enabled (<code>true</code>) or disabled (<code>false</code>).
enableClickjackNonsetupUser	boolean	Indicates whether clickjack protection for non-setup customer pages is enabled (<code>true</code>) or disabled (<code>false</code>).
enableClickjackSetup	boolean	Indicates whether clickjack protection for setup pages is enabled (<code>true</code>) or disabled (<code>false</code>).
enableSMSIdentity	boolean	Indicates whether users can receive a one-time PIN delivered via SMS (<code>true</code>) or not (<code>false</code>).
forceRelogin	boolean	Indicates whether an administrator that is logged in as another user is required to log in again to their original session, after logging out as the secondary user (<code>true</code>) or not (<code>false</code>).
lockSessionsToIp	boolean	Indicates whether user sessions are locked to the IP address from which the user logged in (<code>true</code>) or not (<code>false</code>).
sessionTimeout	SessionTimeout (enumeration of type string)	The length of time after which users without activity are prompted to log out or continue working. Valid values are: <ul style="list-style-type: none"> FifteenMinutes ThirtyMinutes SixtyMinutes TwoHours FourHours

Field	Field Type	Description
		<ul style="list-style-type: none"> • EightHours • TwelveHours

Declarative Metadata Sample Definition

This is a sample security.settings metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<SecuritySettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <networkAccess>
    <ipRanges>
      <end>127.0.0.1</end>
      <start>127.0.0.1</start>
    </ipRanges>
  </networkAccess>
  <passwordPolicies>
    <apiOnlyUserHomePageURL>http://www.altPage.com</apiOnlyUserHomePageURL>
    <complexity>SpecialCharacters</complexity>
    <expiration>OneYear</expiration>
    <passwordAssistanceURL>http://www.acme.com/forgotpassword</passwordAssistanceURL>
    <passwordAssistanceMessage>Forgot your password? Reset it
here.</passwordAssistanceMessage>
    <historyRestriction>3</historyRestriction>
    <lockoutInterval>ThirtyMinutes</lockoutInterval>
    <maxLoginAttempts>ThreeAttempts</maxLoginAttempts>
    <minPasswordLength>TenCharacters</minPasswordLength>
    <questionRestriction>None</questionRestriction>
  </passwordPolicies>
  <sessionSettings>
    <disableTimeoutWarning>true</disableTimeoutWarning>
    <enableCSRFOnGet>false</enableCSRFOnGet>
    <enableCSRFOnPost>false</enableCSRFOnPost>
    <enableCacheAndAutocomplete>false</enableCacheAndAutocomplete>
    <enableClickjackNonsetupSFDC>true</enableClickjackNonsetupSFDC>
    <enableClickjackNonsetupUser>true</enableClickjackNonsetupUser>
    <enableClickjackSetup>true</enableClickjackSetup>
    <enableSMSIdentity>true</enableSMSIdentity>
    <forceReLogin>true</forceReLogin>
    <lockSessionsToIp>true</lockSessionsToIp>
    <sessionTimeout>TwelveHours</sessionTimeout>
  </sessionSettings>
</SecuritySettings>
```

See Also:

[Settings](#)

SharedTo

Use SharedTo to specify the target and source for owner-based sharing rules. See “Sharing Considerations” and “About Groups” in the Database.com online help.

Declarative Metadata File Suffix and Directory Location

SharedTo is used with [OwnerSharingRule](#).

Version

SharedTo is available in API version 17.0 and later.

Fields

Field	Field Type	Description
<code>allInternalUsers</code>	string	A group containing all internal users. This field is available in API version 24.0 and later.
<code>group</code>	string[]	A list of groups with sharing access. Use this field instead of the <code>groups</code> field. This field is available in API version 22.0 and later.
<code>groups</code>	string[]	A list of groups with sharing access. Use the <code>group</code> field instead for API version 22.0 and later.
<code>role</code>	string[]	A list of roles with sharing access. Use this field instead of the <code>roles</code> field. This field is available in API version 22.0 and later.
<code>roleAndSubordinates</code>	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. Use this field instead of the <code>rolesAndSubordinates</code> field. This field is available in API version 22.0 and later.
<code>roleAndSubordinatesInternal</code>	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. This field is available in API version 22.0 and later.
<code>roles</code>	string[]	A list of roles with sharing access. Use the <code>role</code> field instead for API version 22.0 and later.
<code>rolesAndSubordinates</code>	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. Use the <code>roleAndSubordinates</code> field instead for API version 22.0 and later.
<code>queue</code>	string[]	A list of queues with sharing access. Applies only to <code>CustomObject</code> sharing rules. This field is available in API version 24.0 and later.

SharingRules

Represents a set of sharing rules. `SharingRules` enables you to share records with a set of users, using rules that specify the access level of the target user group. It extends the `Metadata` metadata type and inherits its `fullName` field. For more information, see “Sharing Rules Overview” in the Database.com online help.



Note: You can't create a `SharingRules` component directly. Use the types that extend it, such as `CustomObjectSharingRules` instead. This object does not include support for packaging.

Declarative Metadata File Suffix and Directory Location

SharingRules are stored in their corresponding entity directory and the file name matches the entity name. SharingRules for custom objects are stored in the `customObjectSharingRules` directory, which contains files with the `.sharingRules` extension such as `ObjA__c.sharingRules`, where `ObjA` refers to the developer name of a custom object type.

Version

SharingRules components are available in API version 24.0 and later.

Fields

The following information assumes that you are familiar with implementing sharing rules for custom objects. For more information on these fields, see “Overview of Sharing Settings” in the Database.com online help.

Field	Field Type	Description
<code>fullName</code>	<code>string</code>	The unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

CustomObjectSharingRules

Represents the sharing rules for custom objects. It extends the [SharingRules](#) metadata type and inherits its `fullName` field.

Field	Field Type	Description
<code>criteriaBasedRules</code>	CustomObjectCriteriaBasedSharingRule []	List that defines user criteria-based rules.
<code>ownerRules</code>	CustomObjectOwnerSharingRule []	List that defines user membership-based rules.

UserSharingRules

Represents the sharing rules for users. With user sharing rules, you can share members of a group with members of another group. It extends the [SharingRules](#) metadata type and inherits its `fullName` field.

Field	Field Type	Description
<code>criteriaBasedRules</code>	UserCriteriaBasedSharingRule []	List that defines user criteria-based rules.
<code>membershipRules</code>	UserMembershipSharingRule []	List that defines user membership-based rules.

Declarative Metadata Sample Definition

The following is the definition of a user criteria-based sharing rule and a user membership-based sharing rule. The file name corresponds to `User.sharingRules` under the `userSharingRules` directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<UserSharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
  <criteriaBasedRules>
    <fullName>shareUsers2</fullName>
    <sharedTo>
      <group>Asia_Division</group>
    </sharedTo>
    <criteriaItems>
      <field>FirstName</field>
```

```

        <operation>equals</operation>
        <value>John</value>
    </criteriaItems>
    <name>shareUsers2</name>
    <userAccessLevel>Read</userAccessLevel>
</criteriaBasedRules>
<membershipRules>
    <fullName>shareUsers1</fullName>
    <sharedTo>
        <group>South_America_Division</group>
    </sharedTo>
    <sharedFrom>
        <group>Asia_Division</group>
    </sharedFrom>
    <name>shareUsers1</name>
    <userAccessLevel>Read</userAccessLevel>
</membershipRules>
</UserSharingRules>

```

The following shows a sample package.xml file.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>UserRule</members>
        <name>UserCriteriaBasedSharingRule</name>
    </types>
    <types>
        <members>*</members>
        <name>UserMembershipSharingRule</name>
    </types>
    <version>30.0</version>
</Package>

```

BaseSharingRule

Represents the base container for criteria-based and owner-based sharing rules. It extends the [Metadata](#) metadata type and inherits its `fullName` field.



Note: You can't create a `BaseSharingRule` component directly. Use the components under the [CriteriaBasedSharingRule](#) or [OwnerSharingRule](#) metadata types instead.

Version

`BaseSharingRule` components are available in API version 24.0 and later.

Fields

For more information on these fields, see “Overview of Sharing Settings” in the Database.com online help.

Field	Field Type	Description
<code>sharedTo</code>	SharedTo	Required. Specifies who the record should be shared with.
<code>fullName</code>	string	The unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two

Field	Field Type	Description
		consecutive underscores. This field is inherited from the Metadata component.

CriteriaBasedSharingRule

Represents a criteria-based sharing rule. `CriteriaBasedSharingRule` enables you to share records based on specific criteria. It extends the [BaseSharingRule](#) metadata type and inherits its `sharedTo` field. For more information, see “Criteria-Based Sharing Rules Overview” in the Database.com online help.



Note: You can't create a `CriteriaBasedSharingRule` component directly. Use the child components instead.

Declarative Metadata File Suffix and Directory Location

`CriteriaBasedSharingRule` components are stored within the `SharingRules` component in the `criteriaBasedRules` field.

Version

`CriteriaBasedSharingRule` components are available in API version 24.0 and later.

Fields

The following information assumes that you are familiar with implementing sharing rules for standard objects and custom objects. For more information on these fields, see “Overview of Sharing Settings” in the Database.com online help.

Field	Field Type	Description
<code>criteriaItems</code>	FilterItem[]	List that represents the criteria for the sharing rule. The possible values are: <ul style="list-style-type: none"> <code>field</code> <code>operation</code> <code>value</code>

CustomObjectCriteriaBasedSharingRule

Represents a criteria-based sharing rule for custom objects. It extends the [CriteriaBasedSharingRule](#) metadata type and inherits its `criteriaItems` field.

`CustomObjectCriteriaBasedSharingRule` is used by the `criteriaBasedRules` field in [CustomObjectSharingRules](#).

Field	Field Type	Description
<code>accessLevel</code>	string	Required. A value that represents the type of allowed sharing. The possible values are: <ul style="list-style-type: none"> <code>Read</code> <code>Edit</code> <code>All</code>
<code>booleanFilter</code>	string	Represents the filter logic of the sharing rule.

Field	Field Type	Description
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
name	string	Required. Name for the sharing rule. Corresponds to Label in the user interface.

UserCriteriaBasedSharingRule

Represents a criteria-based sharing rule for users. It extends the [CriteriaBasedSharingRule](#) metadata type and inherits its `criteriaItems` field.

`UserCriteriaBasedSharingRule` is used by the `criteriaBasedRules` field in [UserSharingRules](#).

Field	Field Type	Description
booleanFilter	string	Represents the filter logic of the sharing rule.
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
name	string	Required. Name for the sharing rule. Corresponds to Label in the user interface.
userAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the type of allowed sharing. The possible values are: <ul style="list-style-type: none"> • Read • Edit

Declarative Metadata Sample Definition

The following is the definition of two owner-based sharing rules and one criteria-based sharing rule containing two criteria items. The file name corresponds to the `Account.sharingRules` file under the `accountSharingRules` directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountSharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
  <ownerRules>
    <fullName>G1Dev_G2New</fullName>
    <sharedTo>
      <group>G2New</group>
    </sharedTo>
    <sharedFrom>
      <group>G1Dev</group>
    </sharedFrom>
    <accountAccessLevel>Read</accountAccessLevel>
    <caseAccessLevel>None</caseAccessLevel>
    <contactAccessLevel>Read</contactAccessLevel>
  </ownerRules>
  <fullName>G2New_R1New</fullName>
  <sharedTo>
    <roleAndSubordinates>R1New</roleAndSubordinates>
  </sharedTo>
  <sharedFrom>
    <group>G2New</group>
  </sharedFrom>
  <accountAccessLevel>Edit</accountAccessLevel>
  <caseAccessLevel>Read</caseAccessLevel>
  <contactAccessLevel>Edit</contactAccessLevel>
</AccountSharingRules>
```

```

    <name>G2New_R1New</name>
    <opportunityAccessLevel>None</opportunityAccessLevel>
  </ownerRules>
  <criteriaBasedRules>
    <fullName>AccountCriteria</fullName>
    <sharedTo>
      <group>G1</group>
    </sharedTo>
    <criteriaItems>
      <field>BillingCity</field>
      <operation>equals</operation>
      <value>San Francisco</value>
    </criteriaItems>
    <criteriaItems>
      <field>MyChkBox__c</field>
      <operation>notEqual</operation>
      <value>False</value>
    </criteriaItems>
    <accountAccessLevel>Read</accountAccessLevel>
    <booleanFilter>1 OR 2</booleanFilter>
    <caseAccessLevel>None</caseAccessLevel>
    <contactAccessLevel>Read</contactAccessLevel>
    <name>AccountCriteria</name>
    <opportunityAccessLevel>None</opportunityAccessLevel>
  </criteriaBasedRules>
</AccountSharingRules>

```

OwnerSharingRule

Represents an ownership-based sharing rule. OwnerSharingRule enables you to share records owned by a set of users with another set, using rules that specify the access level of the target user group. It extends the [BaseSharingRule](#) metadata type and inherits its SharedTo field. For more information, see “Sharing Rules Overview” in the Database.com online help.



Note: You can't create a OwnerSharingRule component directly. Use the child components instead.

Declarative Metadata File Suffix and Directory Location

OwnerSharingRules components are stored within the SharingRules component in the ownerRules field.

Version

OwnerSharingRules components are available in API version 24.0 and later.

Fields

The following information assumes that you are familiar with implementing sharing rules for standard objects and custom objects. For more information on these fields, see “Overview of Sharing Settings” in the Database.com online help.

Field	Field Type	Description
sharedFrom	SharedTo	Required. Specifies the record owners.
sharedTo	SharedTo	Required. Specifies who the record should be shared with.
fullName	string	The unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not

Field	Field Type	Description
		end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

CustomObjectOwnerSharingRule

Represents a sharing rule for custom objects. It extends the [OwnerSharingRule](#) metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

`CustomObjectOwnerSharingRule` is used by the `ownerRules` field in [CustomObjectSharingRules](#).

Field	Field Type	Description
<code>accessLevel</code>	string	Required. A value that represents the level of access that a group or role is granted to a custom object. The possible values are: <ul style="list-style-type: none"> • Read • Edit • All
<code>description</code>	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
<code>name</code>	string	Required. Name for the sharing rule. Corresponds to Label in the user interface.

UserMembershipSharingRule

Represents a sharing rule to share members of a group with another group of users. It extends the [OwnerSharingRule](#) metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

`UserMembershipSharingRule` is used by the `ownerRules` field in [UserSharingRules](#) on page 164.

Field	Field Type	Description
<code>description</code>	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
<code>name</code>	string	Required. Name for the sharing rule. Corresponds to Label in the user interface.
<code>userAccessLevel</code>	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a user. The possible values are: <ul style="list-style-type: none"> • Read • Edit

SiteDotCom

Represents a site for deployment. It extends the [MetadataWithContent](#) type and inherits its `fullName` and `content` fields.

Declarative Metadata File Suffix and Directory Location

SiteDotCom components are stored in the `siteDotComSites` directory of the corresponding package directory. The file name for the metadata `.xml` file is `[sitename].site-meta.xml`. The file name for the site file is `[sitename].site`



Note: There is a file size limitation when using the Metadata API to deploy a site from test database to production. The assets in the `.site` file can't be larger than 40 MB. The site gets created, but the assets show in the new site as broken. To fix the assets, export the assets from the test database environment separately and then import them into your new site.

Version

SiteDotCom components are available in API version 30.0 and later.

Fields

Field	Field Type	Description
label	string	The name of the site you are deploying.
siteType	(enumeration of type string)	Required. Identifies whether the site is a ChatterNetworkPicasso site for Salesforce Communities sites, or a Siteforce site for Site.com sites.

Declarative Metadata Sample Definition

Sample XML definitions for SiteDotCom are shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<SiteDotCom xmlns="http://soap.sforce.com/2006/04/metadata">
  <label>testsite</label>
  <siteType>Siteforce</siteType>
</SiteDotCom>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SiteDotCom xmlns="http://soap.sforce.com/2006/04/metadata">
  <label>testCommunity</label>
  <siteType>ChatterNetworkPicasso</siteType>
</SiteDotCom>
```

Workflow

Represents the metadata associated with a 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. For more information, see “Workflow Overview” in the Database.com Help. It extends the [Metadata](#) metadata type and inherits its `fullName` field. Use this metadata type to create, update, or delete workflow rule definitions.

When using a manifest file, retrieve all workflow components using the following code:

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


Declarative Metadata File Suffix and Directory Location

Workflow files have the suffix `.workflow`. There is one file per custom object that has workflow. These files are stored in the `workflows` directory of the corresponding package.

Version

Workflow rules are available in API version 13.0 and later.

Workflow

This metadata type represents the valid types of workflow rules and actions associated with a custom object.

Field Name	Field Type	Description
<code>fieldUpdates</code>	WorkflowFieldUpdate []	An array of all field updates for the object associated with the workflow.
<code>fullName</code>	string	The developer name used as a unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
<code>knowledgePublishes</code>	WorkflowKnowledgePublish []	An array of Salesforce Knowledge Workflow Publishes associated with the workflow. Available in API version 27.0 and later.
<code>outboundMessages</code>	WorkflowOutboundMessage []	An array of all of the outbound messages for the object associated with the workflow.
<code>rules</code>	WorkflowRule []	An array of all the objects associated with the workflow.

WorkflowActionReference

`WorkflowActionReference` represents one of the workflow actions.

Field Name	Field Type	Description
<code>name</code>	string	Required. The name of the workflow action.
<code>type</code>	WorkflowActionType (enumeration of type string)	Required. Available types of workflow actions: <ul style="list-style-type: none"> FieldUpdate OutboundMessage

WorkflowFieldUpdate

`WorkflowFieldUpdate` represents a workflow field update. Field updates allow you to automatically update a field value to one that you specify when a workflow rule is triggered. For more information, see “Defining Field Updates” in the Database.com Help.

Field Name	Field Type	Description
<code>description</code>	string	The description of the field update. This information is useful to track the reasoning for initially configuring the field update.
<code>field</code>	string	Required. The field (on the object for the workflow) to be updated.

Field Name	Field Type	Description
formula	string	If the <code>operation</code> field value is <code>Formula</code> , this is set to a formula used to compute the new field value.
fullName	string	Required. The developer name used as a unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the <code>Metadata</code> component.
literalValue	string	If the <code>operation</code> field value is <code>Literal</code> , this is the literal value for the field.
lookupValue	string	If the <code>operation</code> field value is <code>lookupValue</code> , this is the lookup value that is referenced.
lookupValueType	LookupValueType (enumeration of type string)	The type of object that the <code>lookupValue</code> field value is referencing. The valid values are: <ul style="list-style-type: none"> • Queue • RecordType • User
name	string	Required. A name for the component. Available in version API 16.0 and later.
notifyAssignee	boolean	Required. Notify the assignee when the field is updated.
operation	FieldUpdateOperation (enumeration of type string)	Required. The operation that computes the value with which to update the field. Valid values are: <ul style="list-style-type: none"> • <code>Formula</code> - Indicates the field will be set to a formula. If set, the formula must be a valid formula. • <code>Literal</code> - Indicates the field will be set to a literal value. If set, the <code>literalValue</code> must be a valid literal value for this field. • <code>LookupValue</code> - Similar to <code>Literal</code>, but for an object reference, such as a contact, user, account, etc. If set, the <code>lookupValue</code> element must be set. Only <code>User</code> is supported in the current API. • <code>NextValue</code> - Indicates that the field will be set to its next value; this is only allowed when the field update references a picklist. • <code>Null</code> - Indicates the field will be set to null. • <code>PreviousValue</code> - Indicates that the field will be set to its previous value; this is only allowed when the field update references a picklist.
protected	boolean	Required. Indicates whether this component is protected (<code>true</code>) or not (<code>false</code>). Protected components cannot be linked to or referenced by components created in the installing organization.
reevaluateOnChange	boolean	When set to true, if the field update changes the field's value, all workflow rules on the associated object are re-evaluated. Any workflow rules whose criteria are met as a result of the field value change will be triggered.

Field Name	Field Type	Description
		If any of the triggered workflow rules result in another field update that's also enabled for workflow rule re-evaluation, a domino effect occurs, and more workflow rules can be re-evaluated as a result of the newly-triggered field update. This cascade of workflow rule re-evaluation and triggering can happen up to five times after the initial field update that started it.
targetObject	string	This is set if the change is detected on a child record. If this is set, it points to the foreign key reference on the child object (for example, <code>EmailMessage.ParentId</code>) pointing to the parent (for example, <code>Case</code>). When set, the formula is based on the child object (for example, <code>EmailMessage</code>).

WorkflowKnowledgePublish


WorkflowKnowledgePublish represents Salesforce Knowledge article publishing actions and information. Available in API version 27.0 and later.

Field Name	Field Type	Description
action	KnowledgeWorkflowAction (enumeration of type string)	The article publishing actions available when this rule fires. Valid values are: <ul style="list-style-type: none"> <code>PublishAsNew</code>: Publishes the article as a new article. <code>Publish</code>: Publishes the article as a version of a previously published article.
description	string	A brief article description.
label	string	Label that represents the article throughout the Salesforce user interface.
language	string	The language of the article.
protected	boolean	Required. Indicates whether this component is protected (<code>true</code>) or not (<code>false</code>). Protected components cannot be linked to or referenced by components created in the installing organization.

WorkflowOutboundMessage

WorkflowOutboundMessage represents an outbound message associated with a workflow rule. Outbound messages are workflow actions that send 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. For more information, see “Defining Outbound Messages” in the Database.com Help.

Field Name	Field Type	Description
apiVersion	double	Required. The API version of the outbound message. This is automatically set to the current API version when the outbound message is created. Valid API versions for outbound messages are 8.0 and 18.0 or later.

Field Name	Field Type	Description
		<p>This API version is used in API calls back to Database.com using the enterprise or partner WSDLs. The <code>API Version</code> can only be modified by using the Metadata API. It can't be modified using the Database.com user interface. This field is available in API version 18.0 and later.</p> <p> Warning: If you change the <code>apiVersion</code> to a version that doesn't support one of the <code>fields</code> configured for the outbound message, messages will fail until you update your outbound message listener to consume the updated WSDL. You can monitor the status of outbound messages from Setup by clicking Monitoring > Outbound Messages in Database.com.</p>
<code>description</code>	string	Describes the outbound message.
<code>endpointUrl</code>	string	Required. The endpoint URL to which the outbound message is sent.
<code>fields</code>	string[]	The named references to the fields that are to be sent.
<code>fullName</code>	string	Required. The developer name used as a unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
<code>includeSessionId</code>	boolean	Required. Set if you want the Database.com <i>session ID</i> included in the outbound message. Useful if you intend to make API calls and you do not want to include a username and password.
<code>integrationUser</code>	string	Required. The named reference to the user under which this message is sent.
<code>name</code>	string	Required. A name for the component. Available in version API 16.0 and later.
<code>protected</code>	boolean	Required. Indicates whether this component is protected (<code>true</code>) or not (<code>false</code>). Protected components cannot be linked to or referenced by components created in the installing organization.

WorkflowRule

This metadata type represents a workflow rule. It extends the [Metadata](#) metadata type and inherits its `fullName` field.

Field Name	Field Type	Description
<code>actions</code>	WorkflowActionReference []	An array of references for the actions that should happen when this rule fires.
<code>active</code>	boolean	Required. Determines if this rule is active.
<code>booleanFilter</code>	string	For advanced criteria filter, the boolean formula, for example, (1 AND 2) OR 3.

Field Name	Field Type	Description
criteriaItems	FilterItem []	An array of the boolean criteria (conditions) under which this rule fires. Note that either this or <code>formula</code> must be set.
description	string	The description of the workflow rule
formula	string	The formula condition under which this rule first (either this or <code>criteriaItems</code>) must be set
fullName	string	The developer name used as a unique identifier for API access. The <code>fullName</code> can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
triggerType	WorkflowTriggerTypes (enumeration of type string)	Under what conditions the trigger fires. Valid values are: <ul style="list-style-type: none"> <code>onAllChanges</code> - The workflow rule is considered on all changes. <code>onCreateOnly</code> - The workflow rule is considered only on create. <code>onCreateOrTriggeringUpdate</code> - The workflow rule is considered on create and triggering updates.
workflowTimeTriggers	WorkflowTimeTrigger	Represents a set of Workflow actions (Field Updates and Outbound Messages) that should execute before or after a specified interval of time.

WorkflowTimeTrigger

Represents a set of Workflow actions (Field Updates and Outbound Messages) that should execute before or after a specified interval of time.

Field Name	Field Type	Description
actions	WorkflowActionReference []	An array of references for the actions that should happen when this trigger fires.
offsetFromField	string	The date type field name that the time-based workflow triggers off of, i.e. <code>Created Date</code> , <code>Last Modified Date</code> , <code>Rule Trigger Date</code> or a custom date field on the object for which the workflow rule is defined.
timeLength	string	The numeric value of the time after/before the workflow triggers. A negative value represents the time length before the trigger will fire.
workflowTimeTriggerUnit	WorkflowTimeUnits (enumeration of type string)	The unit of time before or after which the time-based workflow will trigger. Valid string values are: <ul style="list-style-type: none"> Hours Days

Declarative Metadata Sample Definition

The following is the definition of a workflow rule:

```
<?xml version="1.0" encoding="UTF-8"?>
<Workflow xmlns="http://soap.sforce.com/2006/04/metadata">
  <alerts>
    <fullName>Another_alert</fullName>
    <description>Another alert</description>
    <protected>>false</protected>
    <recipients>
      <type>accountOwner</type>
    </recipients>
    <recipients>
      <field>Contact__c</field>
      <type>contactLookup</type>
    </recipients>
    <recipients>
      <field>Email__c</field>
      <type>email</type>
    </recipients>
    <template>TestEmail/Email Test</template>
  </alerts>
  <fieldUpdates>
    <fullName>Enum_Field_Update</fullName>
    <description>Blah</description>
    <field>EnumField__c</field>
    <name>Enum Field Update</name>
    <notifyAssignee>>true</notifyAssignee>
    <operation>NextValue</operation>
    <protected>>false</protected>
  </fieldUpdates>
  <fieldUpdates>
    <fullName>Enum_Field_Update2</fullName>
    <description>Blah</description>
    <field>EnumField__c</field>
    <literalValue>PLX2</literalValue>
    <name>Enum Field Update2</name>
    <notifyAssignee>>true</notifyAssignee>
    <operation>Literal</operation>
    <protected>>false</protected>
  </fieldUpdates>
  <fieldUpdates>
    <fullName>Field_Update</fullName>
    <description>TestField update desc</description>
    <field>Name</field>
    <formula>Name & amp; &quot;Updated&quot;</formula>
    <name>Field Update</name>
    <notifyAssignee>>false</notifyAssignee>
    <operation>Formula</operation>
    <protected>>false</protected>
  </fieldUpdates>
  <fieldUpdates>
    <fullName>Lookup_On_Contact</fullName>
    <field>RealOwner__c</field>
    <lookupValue>admin@acme.com</lookupValue>
    <name>Lookup On Contact</name>
    <notifyAssignee>>false</notifyAssignee>
    <operation>LookupValue</operation>
    <protected>>false</protected>
  </fieldUpdates>
  <outboundMessages>
    <fullName>Another_Outbound_message</fullName>
    <description>Another Random outbound.</description>
    <endpointUrl>http://www.test.com</endpointUrl>
    <fields>Email__c</fields>
    <fields>Id</fields>
    <fields>Name</fields>
    <includeSessionId>>true</includeSessionId>
  </outboundMessages>
</Workflow>
```

```

    <integrationUser>admin@acme.com</integrationUser>
    <name>Another Outbound message</name>
    <protected>>false</protected>
  </outboundMessages>
  <rules>
    <fullName>BooleanFilter</fullName>
    <active>>false</active>
    <booleanFilter>1 AND 2 OR 3</booleanFilter>
    <criteriaItems>
      <field>CustomObjectForWorkflow__c.CreatedById</field>
      <operation>notEqual</operation>
    </criteriaItems>
    <criteriaItems>
      <field>CustomObjectForWorkflow__c.CreatedById</field>
      <operation>notEqual</operation>
      <value>abc</value>
    </criteriaItems>
    <criteriaItems>
      <field>CustomObjectForWorkflow__c.CreatedById</field>
      <operation>equals</operation>
      <value>xyz</value>
    </criteriaItems>
    <triggerType>onCreateOrTriggeringUpdate</triggerType>
  </rules>
  <rules>
    <fullName>Custom Rule1</fullName>
    <actions>
      <name>Another_alert</name>
      <type>Alert</type>
    </actions>
    <actions>
      <name>Enum_Field_Update2</name>
      <type>FieldUpdate</type>
    </actions>
    <actions>
      <fullName>Field_Update</fullName>
      <type>FieldUpdate</type>
    </actions>
    <actions>
      <name>Another_Outbound_message</name>
      <type>OutboundMessage</type>
    </actions>
    <actions>
      <name>Role_task_was_completed</name>
      <type>Task</type>
    </actions>
    <active>>true</active>
    <criteriaItems>
      <field>CustomObjectForWorkflow__c.Name</field>
      <operation>startsWith</operation>
      <value>ABC</value>
    </criteriaItems>
    <description>Custom Rule1 desc</description>
    <triggerType>onCreateOrTriggeringUpdate</triggerType>
  </rules>
  <rules>
    <fullName>IsChangedFunctionRule</fullName>
    <active>>true</active>
    <description>IsChangedDesc</description>
    <formula>ISCHANGED(Name)</formula>
    <triggerType>onAllChanges</triggerType>
  </rules>
  <tasks>
    <fullName>Another_task_was_completed</fullName>
    <assignedToType>owner</assignedToType>
    <description>Random Comment</description>
    <dueDateOffset>20</dueDateOffset>
    <notifyAssignee>true</notifyAssignee>
    <priority>High</priority>
    <protected>>false</protected>
  </tasks>

```

```
    <status>Completed</status>
    <subject>Another task was completed</subject>
  </tasks>
  <tasks>
    <fullName>Role_task_was_completed</fullName>
    <assignedTo>R11</assignedTo>
    <assignedToType>role</assignedToType>
    <dueDateOffset>-2</dueDateOffset>
    <notifyAssignee>true</notifyAssignee>
    <offsetFromField>CustomObjectForWorkflow__c.CreatedDate</offsetFromField>
    <priority>High</priority>
    <protected>>false</protected>
    <status>Completed</status>
    <subject>Role task was completed</subject>
  </tasks>
  <tasks>
    <fullName>User_task_was_completed</fullName>
    <assignedTo>admin@acme.com</assignedTo>
    <assignedToType>user</assignedToType>
    <dueDateOffset>-2</dueDateOffset>
    <notifyAssignee>true</notifyAssignee>
    <offsetFromField>User.CreatedDate</offsetFromField>
    <priority>High</priority>
    <protected>>false</protected>
    <status>Completed</status>
    <subject>User task was completed</subject>
  </tasks>
</Workflow>
```


[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 Database.com 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. 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.

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 Database.com. 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 Database.com 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. See also Composite App and Native App.

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.

Controlling Field

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

Custom Field

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

Custom Links

Custom links are URLs defined by administrators to integrate your Database.com 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.

D**Database**

An organized collection of information. The underlying architecture of Database.com 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 Database.com.

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.

Database.com 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 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.

E**Enterprise WSDL**

A strongly-typed WSDL for customers who want to build an integration with their Database.com 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 Database.com 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 Database.com 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.

Filter Condition/Criteria

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

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 Record ID.

Instance

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

Integration User

A Database.com 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**Local Project**

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

Logged-in User

In a SOAP API context, the username used to log into Database.com. 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**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, invoice statements have a master-detail relationship with line items. This type of relationship affects record deletion and security.

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 on AppExchange, similar to a domain name. Database.com automatically prepends your namespace prefix, followed by two underscores (“_”), to all unique component names in your Database.com organization.

O**Object**

An object allows you to store information in your Database.com 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 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 invoice statement may have one or more line items.

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 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 Database.com setup menu. Then you must configure the external endpoint. You can create a listener for the messages using the SOAP API.

Owner

Individual user to which a record is assigned.

P**Picklist**

Selection list of options available for specific fields in a Database.com 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 Database.com 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 Database.com organization that has live users accessing data.

Q**Queue**

A holding area for items before they are processed. Database.com 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.

R**Record**

A single instance of a Database.com object.

Record Name

A standard field on all Database.com objects. A record name can be either free-form text or an autonumber field. `Record Name` does not have to be a unique value.

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.

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.

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).

Salesforce Record ID

A unique 15- or 18-character alphanumeric string that identifies a single record in Database.com.

Salesforce SOA (Service-Oriented Architecture)

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

Session ID

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

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 Database.com organization—without requiring users to log in with a username and password.

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.

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 Database**

A nearly identical copy of a Database.com production organization. You can create a test database for a variety of purposes, such as testing and training, without compromising the data and applications in your production environment.

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.

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.

URL (Uniform Resource Locator)

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

V**Validation Rule**

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

W**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 Database.com organization's information. See also SOAP API and Bulk API.

Workflow Action

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

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.

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 Database.com Enterprise WSDL or Partner WSDL to communicate with Database.com 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

- A**
- Account Team Roles [16](#)
 - AccountSettings components [147](#)
 - ActivitiesSettings component [148](#)
 - ApexClass component [76](#)
 - ApexTrigger component [78](#)
 - API support policy [3](#)
 - AppMenu component [79](#)
- B**
- Backward compatibility [3](#)
 - BaseSharingRule component [165](#)
 - BusinessHoursSettings component [151](#)
- C**
- call deprecation [3](#)
 - CallCenter component [81](#)
 - Calls
 - cancelDeploy [32](#)
 - checkDeployStatus [31](#)
 - checkRetrieveStatus [40](#)
 - checkStatus [55](#)
 - create (asynchronous) [42](#)
 - createMetadata (synchronous) [46](#)
 - delete (asynchronous) [43](#)
 - deleteMetadata (synchronous) [51](#)
 - deploy [25](#)
 - describeMetadata [56](#)
 - listMetadata [56](#), [58](#)
 - readMetadata (synchronous) [48](#)
 - renameMetadata (synchronous) [53](#)
 - retrieve [34](#)
 - update (asynchronous) [44](#)
 - updateMetadata (synchronous) [50](#)
 - cancel deploy call [32](#)
 - checkDeployStatus metadata call [31](#)
 - checkRetrieveStatus metadata call [40](#)
 - checkStatus metadata call [55](#)
 - Components
 - AccountSettings [147](#)
 - ActivitiesSettings [148](#)
 - Activity Settings [148](#)
 - ApexClass [76](#)
 - ApexTrigger [78](#)
 - AppMenu [79](#)
 - BaseSharingRule [165](#)
 - BusinessHoursSettings [151](#)
 - CallCenter [81](#)
 - CriteriaBasedSharingRule [166](#)
 - CustomField [87](#)
 - CustomObject [83](#)
 - Components (*continued*)
 - Dependent Picklist (see Picklist) [96](#)
 - ExternalDataSource [109](#)
 - Group [111](#)
 - InstalledPackage [112](#)
 - Layout [112](#)
 - list of types [73](#)
 - LiveAgentSettings [155](#)
 - Metadata [124](#)
 - MetadataWithContent [124](#)
 - MobileSettings [155](#)
 - NamedFilter [94](#)
 - OwnerSharingRule [168](#)
 - PermissionSet [126](#)
 - Picklist [96](#)
 - Profile [130](#)
 - Queue [137](#)
 - QuickAction [139](#)
 - RemoteSiteSetting [142](#)
 - Role [143](#)
 - SamlSsoConfig [144](#)
 - SecuritySettings [158](#)
 - Settings [146](#)
 - SharedTo [162](#)
 - SharingReason [101](#)
 - SharingRecalculation [102](#)
 - SharingRules [163](#)
 - SiteDotCom [169](#)
 - unsupported [76](#)
 - ValidationRule [103](#)
 - Weblink [104](#)
 - Workflow [170](#)
 - create call (asynchronous) [42](#)
 - createMetadata call (synchronous) [46](#)
 - CriteriaBasedSharingRule component [166](#)
 - CustomField component [87](#)
 - CustomObject
 - Weblink component [104](#)
 - CustomObject component [83](#)
- D**
- delete call (asynchronous) [43](#)
 - deleteMetadata call (synchronous) [51](#)
 - Dependent Picklist [96](#)
 - Deploy [14](#)
 - deploy call
 - running tests [19](#)
 - Deprecated calls [3](#)
 - describeMetadata call [56](#)
 - Developer resources [4](#)
 - Development platforms [3](#)

- E**
- Error handling [24](#)
 - Expiration of session ID [24](#)
 - ExternalDataSource component [109](#)
- F**
- Field types [107](#)
 - File-based metadata [14](#)
- G**
- Group component [111](#)
- I**
- InstalledPackage component [112](#)
- L**
- Layout component [112](#)
 - listMetadata call [56](#)
 - ListMetadataQuery [58](#)
 - LiveAgentSettings components [155](#)
- M**
- Manifest file [14](#), [16](#)
 - Metadata calls [1](#)
 - Metadata component [124](#)
 - Metadata components [75](#)
 - Metadata types [73](#), [75–76](#)
 - MetadataWithContent component [124](#)
 - MobileSettings component [155](#)
- N**
- NamedFilter component [94](#)
- O**
- Opportunity Team Roles [16](#)
 - OwnerSharingRule component [168](#)
- P**
- Package [125](#)
 - Package versions [76](#)
 - package.xml
 - samples [16](#)
 - PackageVersion [76](#)
 - PermissionSet component [126](#)
 - Picklist component [96](#)
 - Prerequisites [5](#)
 - Profile component [130](#)
- Q**
- Queue component [137](#)
 - Quick start
 - Generate WSDLs [5](#)
 - Import WSDLs [6](#)
 - Java sample [7](#)
 - Prerequisites [5](#)
 - QuickAction component [139](#)
- R**
- readMetadata call (synchronous) [48](#)
 - RemoteSiteSetting component [142](#)
 - renameMetadata call (synchronous) [53](#)
 - Retrieve [14](#)
 - retrieve call [34](#)
 - RetrieveRequest [39](#)
 - Role component [143](#)
- S**
- SamlSsoConfig component [144](#)
 - Sample code [7](#)
 - SecuritySettings component [158](#)
 - Session ID expiration [24](#)
 - Settings [146](#)
 - SharedTo component [162](#)
 - SharingReason component [101](#)
 - SharingRecalculation component [102](#)
 - SharingRules [163](#)
 - SiteDotCom component [169](#)
 - Standards compliance [3](#)
 - Support policy [3](#)
 - Supported calls [75](#)
- T**
- Types of fields [107](#)
- U**
- Understanding metadata calls and components [1](#)
 - update call (asynchronous) [44](#)
 - updateMetadata call (synchronous) [50](#)
 - Usernames [20](#)
- V**
- ValidationRule component [103](#)
 - Versions [76](#)
- W**
- Weblink component [104](#)
 - Workflow component [170](#)

WSC [6](#)
WSDL integration [5-6](#)

Z

Zip file [14](#)