Metadata API Developer Guide

Version 36.0, Spring '16

@salesforcedocs

Last updated: January 20, 2016

CONTENTS

GETTING STARTED

CHAPTER 1 Understanding Metadata API

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. Both tools are built on top of Metadata API and use the standard Eclipse and Ant tools respectively to simplify working with Metadata API.

- Force.com IDE is built on the Eclipse platform, for programmers familiar with integrated development environments. Code, compile, test, and deploy from within the IDE.
- The Force.com Migration Tool is ideal if you use a script or the command line for moving metadata between a local directory and a Salesforce organization.

For more information about the Force.com IDE or Force.com Migration Tool, see developer.salesforce.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 test organizations on Developer Edition or sandbox, and then deploy tested changes to production organizations on Enterprise, Unlimited, or Performance Editions. 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

Supported Salesforce Editions

To use Metadata API, your organization must use **Enterprise Edition**, **Unlimited Edition**, **Performance Edition**, or **Developer Edition**. If you are an existing Salesforce customer and want to upgrade to Enterprise, Unlimited, or Performance Edition, contact your account representative.

It is strongly recommended that you use a sandbox, which is an exact replica of your production organization. Enterprise, Unlimited, and Performance Editions come with a free developer sandbox. For more information, see `http://www.salesforce.com/platform/cloud-infrastructure/sandbox.jsp`.

Alternatively, you can use a Developer Edition organization, which provides access to all of the features available with Enterprise Edition, but is limited by the number of users and the amount of storage space. A Developer Edition organization is not a copy of your production organization, but it provides an environment where you can build and test your solutions without affecting your organization's data. Developer Edition accounts are available for free at `http://developer.salesforce.com/signup`.

Note: A metadata component must be visible in the organization for Metadata API to act on it. Also, a user must have the "API Enabled" permission to have access to metadata components.

Metadata API Access for Professional Edition

ISV partners can request Metadata API access to Professional Edition organizations for apps that have passed the AppExchange Security Review. Access is granted through an API token (client ID). This special key enables the app to make Metadata API calls to customers' Professional Edition organizations.

As an ISV partner, you can request Metadata API access by following these steps.

1. Submit your app for security review. See Steps in the Security Review in the *ISVForce Guide*.
2. After your app is approved, log a case in the Partner Community in **AppExchange and Feature Requests** > **API Token Request**, and specify SOAP for the type of token.

To make calls to the Metadata API, append the API token to the CallOptions SOAP header in your calls.

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()` operations 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 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*.

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

Each new Salesforce release consists of two components:

- A new release of platform software that resides on Salesforce 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 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 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 will directly notify customers using API versions planned for deprecation.

Related Resources

The Salesforce 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://developer.salesforce.com/page/Getting_Started` for more information, or visit `http://developer.salesforce.com/signup` to sign up for a free Developer Edition account.

You can visit these websites to find out more about Salesforce applications:

- Salesforce Developers provides a wealth of information for developers.
- Salesforce for information about the Salesforce application.
- Force.com AppExchange for access to apps created for Salesforce.
- Salesforce.com Community for services to ensure Salesforce customer success.

CHAPTER 2 Quick Start

Use Metadata API to retrieve, deploy, create, update, or delete customizations for your organization. The most common use is to migrate changes from a sandbox or testing organization to your production environment. Metadata API is intended for managing customizations and for building tools that can manage the metadata model, not the data itself.

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 for CRUD-Based Development with Synchronous Calls.

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 sandbox, which is an exact replica of your production organization. Enterprise, Unlimited, and Performance Editions come with a free developer sandbox. For more information, see `http://www.salesforce.com/platform/cloud-infrastructure/sandbox.jsp`.

  Alternatively, you can use a Developer Edition organization, which provides access to all of the features available with Enterprise Edition, but is limited by the number of users and the amount of storage space. A Developer Edition organization is not a copy of your production organization, but it provides an environment where you can build and test your solutions without affecting your organization's data. Developer Edition accounts are available for free at `http://developer.salesforce.com/signup`.

- 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 Salesforce administrator, or you can generate it yourself if you have access to the WSDL download page in the Salesforce 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 Salesforce platform. (The System Administrator profile has this permission.)

The sample code in Step 3: Walk Through the Java Sample Code on page 6 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 Salesforce account. You must log in as an administrator or as a user who has the "Modify All Data" permission.
2. From Setup, enter *API* in the `Quick Find` box, then select **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

When you have imported the WSDL files, you can build 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 sandbox, 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 Salesforce 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/36.0";
        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 a retrieve call, the sample calls `checkRetrieveStatus()` in a loop until the operation is completed. Similarly, after a deploy call, the sample checks `checkDeployStatus()` in a loop until the operation is 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>36.0</version>
</Package>
```

Note the error handling code that follows each API call.

Note: This sample requires API version 34.0 or later.

```
import java.io.*;
import java.nio.channels.Channels;
import java.nio.channels.FileChannel;
import java.nio.channels.ReadableByteChannel;
import java.rmi.RemoteException;
```

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

```
    private void retrieveZip() throws Exception {
        RetrieveRequest retrieveRequest = new RetrieveRequest();
        // The version in package.xml overrides the version in RetrieveRequest
        retrieveRequest.setApiVersion(API_VERSION);
        setUnpackaged(retrieveRequest);

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

        if (result.getStatus() == RetrieveStatus.Failed) {
            throw new Exception(result.getErrorStatusCode() + " msg: " +
                    result.getErrorMessage());
        } else if (result.getStatus() == RetrieveStatus.Succeeded) {
         // 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 RetrieveResult waitForRetrieveCompletion(AsyncResult asyncResult) throws
Exception {
     // Wait for the retrieve to complete
        int poll = 0;
        long waitTimeMilliSecs = ONE_SECOND;
        String asyncResultId = asyncResult.getId();
        RetrieveResult result = null;
        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.");
            }
            result = metadataConnection.checkRetrieveStatus(
                    asyncResultId, true);
            System.out.println("Retrieve Status: " + result.getStatus());
        } while (!result.isDone());

        return result;
    }

    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 quualified 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;
    }
}
```

CHAPTER 3 Deploying and Retrieving Metadata

Use the `deploy()` and `retrieve()` calls to move metadata (XML files) between a Salesforce 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 Salesforce 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 sandbox organization. After development and testing is completed, the application or customization is then deployed into a production organization using Metadata API.
- Team development of an application in a Developer Edition organization. After development and testing is completed, you can then distribute the application via Force.com AppExchange.

SEE ALSO:

Metadata Components and Types

Unsupported Metadata 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 that are 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 that are retrieved or deployed in a .zip file might 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 10,000 files or 400 MB at one time. If either of these limits is exceeded, the deployment or retrieval fails.

Every .zip file contains a project manifest, a file that's named `package.xml`, and a set of directories that contain the components. The manifest file defines the components that you're trying to retrieve or deploy in the .zip file and the API version that's used for the deployment or retrieval.

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>36.0</version>
</Package>
```

The following elements can 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 for determining 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 that's entered must match a metadata type that's defined in the Metadata API WSDL. See Metadata Types for a list.
- `<version>` is the API version number that's used when the .zip file is deployed or retrieved. Currently the valid value is `36.0`.

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 components, see Deleting Components from an Organization.

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:

- Standard Objects
- All Custom Objects
- Standard Picklist Fields
- Custom and Standard Fields
- List Views for Standard Objects
- Packages
- Security Settings
- Assignment Rules, Auto-response Rules, Escalation Rules
- Sharing Rules
- Managed Component Access

Standard Objects

This sample `package.xml` manifest file illustrates how to work with the standard Account object. Retrieving or deploying a standard object includes all custom and standard fields except for standard fields that aren't customizable. All custom fields are supported. Only standard fields that you can customize are supported, that is, standard fields to which you can add help text or enable history tracking or Chatter feed tracking. Other standard fields aren't supported, including system fields (such as `CreatedById` or `LastModifiedDate`) and autonumber fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account</members>
        <name>CustomObject</name>
    </types>
    <version>36.0</version>
</Package>
```

Note how you work with the standard Account object by specifying it as a member of a CustomObject type. However, you cannot use an asterisk wildcard to work with all standard objects; each standard object must be specified by name.

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>36.0</version>
</Package>
```

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

Standard Picklist Fields

This sample `package.xml` manifest file illustrates how to work with the standard `Industry` Picklist field in the standard Account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account.Industry</members>
        <name>CustomField</name>
    </types>
    <version>36.0</version>
</Package>
```

Note the ***objectName.picklistField*** syntax in the `<members>` field where *objectName* is the name of the object, such as `Account`, and *picklistField* is the name of the standard picklist field, such as `Industry`.

This sample `package.xml` manifest file illustrates how to get the opportunity team roles (the same picklist values as the account team roles) using the `TeamMemberRole` picklist field values on the OpportunityTeamMember object. You need to have team selling enabled in your organization to retrieve these values.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>OpportunityTeamMember.TeamMemberRole</members>
        <name>CustomField</name>
    </types>
    <version>36.0</version>
</Package>
```

Custom and Standard Fields

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

```
<?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>
        <members>Account.Phone</members>
        <name>CustomField</name>
    </types>
    <version>36.0</version>
</Package>
```

Note the ***objectName.field*** syntax in the `<members>` field where *objectName* is the name of the object, such as Account, and *field* is the name of the custom or standard 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`. Similarly, the `Phone` standard field in the Account standard object is uniquely identified by its full name, `Account.Phone`.

All custom fields are supported. Only standard fields that you can customize are supported, that is, standard fields to which you can add help text or enable history tracking or Chatter feed tracking. Other standard fields aren't supported, including system fields (such as `CreatedById` or `LastModifiedDate`) and autonumber fields.

List Views for Standard Objects

The easiest way to retrieve list views for a standard object is to retrieve the object. The list views are included in the retrieved component. See Standard Objects on page 16.

You can also work with individual list views if you do not want to retrieve all the details for the object. This sample `package.xml` manifest file illustrates how to work with a list view for the standard Account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account.AccountTeam</members>
        <name>ListView</name>
    </types>
    <version>36.0</version>
</Package>
```

Note the ***objectName.listViewUniqueName*** syntax in the `<members>` field where *objectName* is the name of the object, such as Account, and *listViewUniqueName* is the `View Unique Name` for the list view. If you retrieve this list view, the component is stored in `objects/Account.object`.

Packages

To retrieve a package, set the name of the package in the `packageNames` 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 `packageNames` 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>36.0</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 1 to 15-character alphanumeric identifier that distinguishes your package and its contents from other publishers' packages. For more information, see "Register a Namespace Prefix" in the Salesforce 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>36.0</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>36.0</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>36.0</version>
</Package>
```

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

Sharing Rules

In API version 33.0 and later, you can retrieve and deploy sharing rules for all standard and custom objects. This sample `package.xml` manifest file illustrates how to work with an organization's sharing rules, which includes retrieving a specific criteria-based sharing rule for the lead object, retrieving all ownership-based sharing rules for all objects, and retrieving all territory-based sharing rules for the account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Lead.testShareRule</members>
         <name>SharingCriteriaRule</name>
    </types>
    <types>
        <members>*</members>
        <name>SharingOwnerRule</name>
    </types>
    <types>
        <members>Account.*</members>
        <name>SharingTerritoryRule</name>
    </types>
    <version>33.0</version>
</Package>
```

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
- Apps
- Custom field permissions
- Custom object permissions
- Custom tab settings
- External data sources
- Record types
- Visualforce pages

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>
```

When retrieving permission sets and profiles, make sure that you also retrieve any components that are related to the permissions and settings. For example, when retrieving app visibilities, you must also retrieve the associated app, and when retrieving object or field permissions, you must also retrieve the associated object.

Running Tests in a Deployment

Default Test Execution in Production

When no test level is specified in the deployment options, the default test execution behavior depends on the contents of your deployment package. When deploying to production, all tests, except those that originate from managed packages, are executed if your deployment package contains Apex classes or triggers. If your package doesn't contain Apex components, no tests are run by default.

In API version 33.0 and earlier, tests were run for components that required tests, such as custom objects, and not only for Apex components. For example, if your package contains a custom object, all tests are run in API version 33.0 and earlier. In contrast, starting with API version 34.0, no tests are run for this package. The API version corresponds to the version of your API client or the version of the tool you're using (Force.com Migration Tool).

You can run tests for a deployment of non-Apex components. You can override the default test execution behavior by setting the test level in your deployment options. Test levels are enforced regardless of the types of components present in your deployment package. We recommend that you run all local tests in your development environment, such as sandbox, prior to deploying to production. Running tests in your development environment reduces the amount of tests needed to run in a production deployment.

Note: This default test execution behavior doesn't apply to change sets. When you deploy change sets to production, all local tests are executed if your change sets include components that require tests. The executed tests must cover your code at a minimum of 75%. For a list of components that require tests, see this next section.

Default Test Execution in Production for API Version 33.0 and Earlier

For deployment to a production organization, all local tests in your organization are run by default. Tests that originate from installed managed packages aren't run by default. If any test fails, the entire deployment is rolled back.

If the deployment includes components for the following metadata types, all local tests are run. For change sets, this test execution behavior applies to all API versions.

- 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:

- 1 CustomApplication component
- 100 Report components and 40 Dashboard components

All tests are run for the following deployments:

- 1 CustomField component
- 1 ApexComponent component and 1 ApexClass component
- 5 CustomField components and 1 ApexPage component
- 100 Report components, 40 Dashboard components, and 1 CustomField component

SEE ALSO:

deploy()

Running a Subset of Tests in a Deployment

Test levels enable you to have more control over which tests are run in a deployment. To shorten deployment time to production, run a subset of tests when deploying Apex components. The default test execution behavior in production has also changed. By default, if no test level is specified, no tests are executed, unless your deployment package contains Apex classes or triggers.

If the code coverage of an Apex component in the deployment is less than 75%, the deployment fails. If one of the specified tests fails, the deployment also fails. We recommend that you test your deployment in sandbox first to ensure that the specified tests cover each component sufficiently. Even if your organization's overall code coverage is 75% or more, the individual coverage of the Apex components being deployed can be less. If the code coverage requirement isn't met, write more tests and include them in the deployment.

To run a subset of tests, set the `RunSpecifiedTests` test level on the `DeployOptions` object. Next, specify each test class to run in `DeployOptions`. Finally, pass `DeployOptions` as an argument to the `deploy()` call. The following example performs those steps to run only the specified test classes.

```
// Create the DeployOptions object.
DeployOptions deployOptions = new DeployOptions();

// Set the appropriate test level.
deployOptions.setTestLevel(TestLevel.RunSpecifiedTests);

// Specify the test classes to run.
// String array contains test class names.
String[] tests = {"TestClass1", "TestClass2", "TestClass3"};
// Add the test class names array to the deployment options.
deployOptions.setRunTests(tests);

// Call deploy() by passing the deployment options object as an argument.
AsyncResult asyncResult = metadatabinding.deploy(zipBytes,deployOptions);
```

Notes About Running Specific Tests

- You can only specify test classes. You can't specify individual test methods.
- We recommend that you refactor test classes to include the minimum number of tests that meet code coverage requirements. Refactoring your test classes can contribute to shorter test execution times, and as a result, shorter deployment times.
- You can deactivate a trigger in the target organization by deploying it with an inactive state. However, the trigger must have been previously deployed with an active state.

Run the Same Tests in Sandbox and Production Deployments

Starting in API version 34.0, you can choose which tests to run in your development environment, such as only local tests, to match the tests run in production. In earlier versions, if you enabled tests in your sandbox deployment, you couldn't exclude managed package tests.

By default, no tests are run in a deployment to a non-production organization, such as a sandbox or a Developer Edition organization. To specify tests to run in your development environment, set a `testLevel` deployment option. For example, to run local tests in a deployment and to exclude managed package tests, set `testLevel` on the `DeployOptions` object to `TestLevel.RunLocalTests`. Next, pass this object as an argument to the `deploy()` call as follows.

```
// Create the DeployOptions object.
DeployOptions deployOptions = new DeployOptions();

// Set the appropriate test level.
deployOptions.setTestLevel(TestLevel.RunLocalTests);

// Call deploy() by passing the deployment options object as an argument.
AsyncResult asyncResult = metadatabinding.deploy(zipBytes,deployOptions);
```

Note: The `RunLocalTests` test level is enforced regardless of the contents of the deployment package. In contrast, tests are executed by default in production only if your deployment package contains Apex classes or triggers. You can use `RunLocalTests` for sandbox and production deployments.

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 Salesforce attempts to locate a matching user in the destination organization by comparing usernames during the deployment.

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

For user references in deployments, Salesforce performs the following sequence:

1. Salesforce compares usernames in the source environment to the destination environment and adapts the organization domain name.
2. If two or more usernames match, Salesforce 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, Salesforce 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 Salesforce 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 Salesforce, from Setup, enter *API* in the `Quick Find` box, then select **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 synchronous or asynchronous. CRUD calls are synchronous in API version 30.0 and later, and similar to the API core calls the results are returned in a single call. In earlier API versions, create, update, and delete are only asynchronous, which means that the results are not immediately returned in one call.
- There are synchronous metadata calls that map to the corresponding core SOAP API synchronous calls.
  - `createMetadata()` maps to the `create()` SOAP API call.
  - `updateMetadata()` maps to the `update()` SOAP API call.
  - `deleteMetadata()` maps to the `delete()` SOAP API call.

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 for CRUD-Based Development with Synchronous Calls

This section guides you 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 `createMetadata()` to create a custom object. This call returns the result in one call.
3. Inspects the returned `SaveResult` object to check if the operation succeeded, and if it didn't, writes the component name, error message, and status code to the output.

```
import com.sforce.soap.metadata.*;

/**
 * Sample that logs in and creates a custom object through the metadata API
 */
public class CRUDSampleCreate {
```

```
    private MetadataConnection metadataConnection;

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

    public CRUDSampleCreate() {
    }

    public static void main(String[] args) throws Exception {
        CRUDSampleCreate crudSample = new CRUDSampleCreate();
        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 createCustomObjectSync(final String uniqueName) throws Exception {
        final String label = "My Custom Object";
        CustomObject co = new CustomObject();
        co.setFullName(uniqueName);
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Created by the Metadata API Sample");
        co.setEnableActivities(true);
        co.setLabel(label);
        co.setPluralLabel(label + "s");
        co.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);

        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());
                }
            }
```

```
        }
    }

    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";
        createCustomObjectSync(uniqueObjectName);
    }
}
```

Java Sample for CRUD-Based Development with Asynchronous Calls

Important: The sample in this section depends on the asynchronous `create()` CRUD call. Asynchronous CRUD calls are no longer available as of API version 31.0 and are available only in earlier API versions.

This section guides you through a sample Java client application that uses asynchronous 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.

   Salesforce 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 6.

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 Salesforce 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*.

CHAPTER 6 File-Based Calls

Use the following file-based calls to deploy or retrieve XML components.

- `deploy()`
- `deployRecentValidation()`
- `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 10,000 files or 400 MB at one time. If either of these limits is exceeded, the deployment or retrieval fails.

In API version 29.0, Salesforce improved the deployment status properties and removed the requirement to use `checkStatus()` after a `deploy()` call to get information about deployments. Salesforce 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.

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 `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.
3. Issue a `checkDeployStatus()` call to obtain the results of the `deploy()` call, using the `id` value returned in the first step.

To track the status of deployments that are in progress or completed in the last 30 days, from Setup, enter *Deployment Status* in the `Quick Find` box, then select **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.

The `package.xml` file is a project manifest that lists all the components that you want to retrieve or deploy. You can use `package.xml` to add components. To delete components, add another manifest file. See Deleting Components from an Organization.

Permissions

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

Arguments

Name	Type	Description
`zipFile`	base64	Base 64-encoded binary data. Client applications must encode the binary data as base64.
`deployOptions`	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
`allowMissingFiles`	boolean	Specifies whether a deploy succeeds even if files that are specified in `package.xml` but are not in the `.zip` file (`true` or not `false`). Do not set this argument for deployment to *production organizations*.
`autoUpdatePackage`	boolean	If a file is in the `.zip` file but not specified in `package.xml`, specifies whether the file should be automatically added to the package (`true` or not `false`). A `retrieve()` is automatically issued with the updated `package.xml` that includes the `.zip` file. Do not set this argument for deployment to *production organizations*.

Name	Type	Description
`checkOnly`	boolean	Indicates whether Apex classes and triggers are saved to the organization as part of the deployment (`false`) or not (`true`). Defaults to `false`. Any errors or messages that would have been issued are still generated. This parameter is similar to the Salesforce Ant tool's `checkOnly` parameter.
`ignoreWarnings`	boolean	Indicates whether a warning should allow a deployment to complete successfully (`true`) or not (`false`). Defaults to `false`. The DeployMessage object for a warning contains the following values: • `problemType`—`Warning` • `problem`—The text of the warning. If a warning occurs and `ignoreWarnings` is set to `true`, the `success` field in DeployMessage is `true`. If `ignoreWarnings` is set to `false`, `success` is set to `false` and the warning is treated like an 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.
`performRetrieve`	boolean	Indicates whether a `retrieve()` call is performed immediately after the deployment (`true`) or not (`false`). Set to `true` in order to retrieve whatever was just deployed.
`purgeOnDelete`	boolean	If `true`, the deleted components in the `destructiveChanges.xml` manifest file aren't stored in the Recycle Bin. Instead, they become immediately eligible for deletion. This field is available in API version 22.0 and later. This option only works in Developer Edition or sandbox organizations; it doesn't work in production organizations.
`rollbackOnError`	boolean	Indicates whether any failure causes a complete rollback (`true`) or not (`false`). If `false`, whatever set of actions can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to `true` if you are deploying to a production organization. The default is `false`.
`runAllTests`	boolean	(Deprecated and only available in API version 33.0 and earlier.) This field defaults to `false`. Set to `true` to run all Apex tests after deployment, including tests that originate from installed managed packages.

Name	Type	Description
		Note: Apex tests that run as part of a deployment always run synchronously and serially.
`runTests`	string[]	A list of Apex tests to run during deployment. Specify the class name, one name per instance. The class name can also specify a namespace with a dot notation. For more information, see Running a Subset of Tests in a Deployment. To use this option, set `testLevel` to `RunSpecifiedTests`.
`singlePackage`	boolean	Indicates whether the specified `.zip` file points to a directory structure with a single package (`true`) or a set of packages (`false`).
`testLevel`	TestLevel (enumeration of type string)	Optional. Specifies which tests are run as part of a deployment. The test level is enforced regardless of the types of components that are present in the deployment package. Valid values are: • `NoTestRun`—No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial organizations. This test level is the default for development environments. • `RunSpecifiedTests`—Only the tests that you specify in the `runTests` option are run. Code coverage requirements differ from the default coverage requirements when using this test level. Each class and trigger in the deployment package must be covered by the executed tests for a minimum of 75% code coverage. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage. • `RunLocalTests`—All tests in your organization are run, except the ones that originate from installed managed packages. This test level is the default for production deployments that include Apex classes or triggers. • `RunAllTestsInOrg`—All tests are run. The tests include all tests in your organization, including tests of managed packages. If you don't specify a test level, the default test execution behavior is used. See Running Tests in a Deployment. Note: Apex tests that run as part of a deployment always run synchronously and serially. This field is available in API version 34.0 and later.

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.enterprise.LoginResult;
import com.sforce.soap.enterprise.EnterpriseConnection;
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/c/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 EnterpriseConnection(loginConfig)).login(
                username, password);

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

}
```

IN THIS SECTION:

SEE ALSO:

Running Tests in a Deployment

Deleting Components from an Organization

To delete components, perform a deployment with the `deploy()` call by using a destructive changes manifest file that lists the components to remove from your organization. You can perform a deployment that only deletes components, or a deployment that deletes and adds components. In API version 33.0 and later, you can specify components to delete before and after other components are added or updated. In earlier API versions, if deletions and additions are specified for the same deployment, the `deploy()` call performs the deletions first.

Deleting Components in a Deployment

To delete components, use the same procedure as with deploying components, but also include a delete manifest file that's named `destructiveChanges.xml` and list the components to delete in this manifest. The format of this manifest is the same as `package.xml` except that wildcards aren't supported.

The following sample `destructiveChanges.xml` file 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>
```

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>36.0</version>
</Package>
```

Note:

- To bypass the Recycle Bin, set the `purgeOnDelete` option to `true`.
- If you try to delete some components that don't exist in the organization, the rest of the deletions are still attempted.

Adding and Deleting Components in a Single Deployment

You can perform a deployment that specifies components to delete in `destructiveChanges.xml` and components to add or update in `package.xml`. The process is the same as with performing a delete-only deployment except that `package.xml` contains the components to add or update.

By default, deletions are processed before component additions. In API version 33.0 and later, you can specify components to be deleted before and after component additions. The process is the same as with performing a delete-only deployment except that the name of the deletion manifest file is different.

- To delete components *before* adding or updating other components, create a manifest file that's named `destructiveChangesPre.xml` and include the components to delete.
- To delete components *after* adding or updating other components, create a manifest file that's named `destructiveChangesPost.xml` and include the components to delete.

The ability to specify when deletions are processed is useful when you're deleting components with dependencies. For example, if a custom object is referenced in an Apex class, you can't delete it unless you modify the Apex class first to remove the dependency on the custom object. In this example, you can perform a single deployment that updates the Apex class to clear the dependency and then deletes the custom object by using `destructiveChangesPost.xml`. The following are samples of the `package.xml` and `destructiveChangesPost.xml` manifests that would be used in this example.

Sample `package.xml`, which specifies the class to update:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>SampleClass</members>
        <name>ApexClass</name>
    </types>
    <version>36.0</version>
</Package>
```

Sample `destructiveChangesPost.xml`, which specifies the custom object to delete after the class update:

```
<?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>
```

Note: The API version that the deployment uses is the API version that's specified in `package.xml`.

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, Salesforce improved the deployment status properties and removed the requirement to use `checkStatus()` after a `deploy()` call to get information about deployments. Salesforce 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
`id`	ID	ID obtained from an AsyncResult object returned by `deploy()` or a subsequent `checkDeployStatus()` call.
`includeDetails`	boolean	Sets the DeployResult object to include DeployDetails information ((`true`) or not (`false`). The default is `false`. Available in API version 29.0 and later.

Response

DeployResult

`cancelDeploy()`

Cancels 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 Salesforce user interface from Setup by entering *`Deployment Status`* in the `Quick Find` box, selecting **Deployment Status**, and then 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
`id`	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());
    }
}
```

deployRecentValidation()

Deploys a recently validated component set without running Apex tests.

Syntax

```
string = metadatabinding.deployRecentValidation(ID validationID)
```

Usage

Use `deployRecentValidation()` to deploy your components to production in less time by skipping the execution of Apex tests. Ensure that the following requirements are met before deploying a recent validation.

- The components have been validated successfully for the target environment within the last four days (96 hours).
- As part of the validation, Apex tests in the target org have passed.
- Code coverage requirements are met.
  - If all tests in the org or all local tests are run, overall code coverage is at least 75%, and Apex triggers have some coverage.
  - If specific tests are run with the `RunSpecifiedTests` test level, each class and trigger that was deployed is covered by at least 75% individually.

This call is equivalent to performing a quick deployment of a recent validation on the Deployment Status page in the Salesforce user interface.

Before you call `deployRecentValidation()`, your organization must have a validation that was recently run. You can run a validation on a set of components by calling deploy() with the `checkOnly` property of the *`deployOptions`* parameter set to `true`. Note the ID that you obtained from the `deploy()` call. You'll use this ID for the `deployRecentValidation()` call in the next step.

After you've run a validation successfully, use these steps to quick-deploy the validation to the same target environment.

1. To start an asynchronous quick deployment, call `deployRecentValidation()` and pass it the ID of a recent validation. This ID is obtained from the previous deploy() call. The `deployRecentValidation()` call returns the ID of the quick deployment. Note this value. You'll use it in the next step.
2. Check for the completion of the call. This process is similar to that of deploy(). 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 that was started by using the `deployRecentValidation()` call. When calling `checkDeployStatus()`, pass in the ID value that you obtained in the first step.

Version

Available in API version 33.0 and later.

Arguments

Name	Type	Description
`validationID`	string	The ID of a recent validation.

Response

Type: string

The ID of the quick deployment.

Sample Code—Java

```
package com.salesforce.test.metadata;

import java.rmi.RemoteException;

import com.sforce.soap.metadata.CodeCoverageWarning;
```

```
import com.sforce.soap.metadata.DeployDetails;
import com.sforce.soap.metadata.DeployMessage;
import com.sforce.soap.metadata.DeployResult;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.soap.metadata.RunTestFailure;
import com.sforce.soap.metadata.RunTestsResult;
import com.sforce.soap.partner.Connector;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;

/**
 * Quick-deploy a recent validation.
 * Prerequisite: A successful validation (check-only deploy) has been done in the org
recently.
 */
public class DeployRecentValidationSample {
    // binding for the metadata WSDL used for making metadata API calls
    private MetadataConnection metadataConnection;

    // 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 = args[0];
        final String PASSWORD = args[1];
        final String URL = args[2];

        final String recentValidationId = args[3];

        DeployRecentValidationSample sample = new DeployRecentValidationSample(
                        USERNAME, PASSWORD, URL);
        sample.deployRecentValidation(recentValidationId);
    }

   public DeployRecentValidationSample(String username, String password, String loginUrl)

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

    public void deployRecentValidation(String recentValidationId)
        throws RemoteException, Exception
    {
       String asyncResultId = metadataConnection.deployRecentValidation(recentValidationId);


        // 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 recent validation " + recentValidationId +
                           " was successfully deployed");
    }

    /**
     * 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.setUsername(username);
        loginConfig.setPassword(password);
        loginConfig.setAuthEndpoint(loginUrl);

        Connector.newConnection(loginConfig);


        final ConnectorConfig metadataConfig = new ConnectorConfig();
        metadataConfig.setServiceEndpoint(
            loginConfig.getServiceEndpoint().replace("/u/", "/m/"));
        metadataConfig.setSessionId(loginConfig.getSessionId());
```

```
        this.metadataConnection = com.sforce.soap.metadata.Connector.
            newConnection(metadataConfig);
    }

}
```

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 10,000 files or 400 MB at one time. If either of these limits is exceeded, the deployment or retrieval fails.

In API version 31.0 and later, the process of making a `retrieve()` call has been simplified. You no longer have to call `checkStatus()` after a `retrieve()` call to obtain the status of the retrieve operation. Instead, make calls to `checkRetrieveStatus()` only. If the retrieve operation is in progress, call `checkRetrieveStatus()` again until the retrieve operation is completed. The `checkStatus()` call is still supported in versions API version 30.0 or earlier, but is not available in API version 31.0 and later.

For API version 31.0 or later, retrieve packaged or unpackaged components by using the following steps.

1. Issue a `retrieve()` call to start the asynchronous retrieval. An AsyncResult object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkRetrieveStatus()` call and pass in the `id` value from the AsyncResult object from the first step. Check the value of the `done` field of the returned RetrieveResult. If it is `true`, this means that the call is completed and proceed to the next step. Otherwise, repeat this step to call `checkRetrieveStatus()` again until the `done` field is `true`.
3. Retrieve the zip file (`zipFile` field) and other desired fields from RetrieveResult that was returned by the final call to `checkRetrieveStatus()` in the previous step.

For API version 30.0 or earlier, retrieve packaged or unpackaged components by using the following steps.

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.

For examples of manifest files, see Sample `package.xml` Manifest Files.

Permissions

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

Arguments

Name	Type	Description
`retrieveRequest`	RetrieveRequest	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 requires API version 34.0 or later.

```
package com.doc.samples;

import java.io.*;
import java.util.*;
import java.nio.ByteBuffer;
import java.nio.channels.*;
import java.rmi.RemoteException;
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.metadata.AsyncResult;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.soap.enterprise.EnterpriseConnection;
import com.sforce.soap.metadata.RetrieveMessage;
import com.sforce.soap.metadata.RetrieveRequest;
import com.sforce.soap.metadata.RetrieveResult;
import com.sforce.soap.metadata.RetrieveStatus;
import com.sforce.soap.enterprise.LoginResult;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;
import com.sforce.soap.metadata.PackageTypeMembers;

public class RetrieveSample {

    // Binding for the metadata WSDL used for making metadata API calls
    private MetadataConnection metadataConnection;
```

```
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 = 31.0;

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/c/31.0";

    RetrieveSample sample = new RetrieveSample(USERNAME, PASSWORD, URL);
    sample.retrieveZip();
}

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


private void retrieveZip() throws RemoteException, Exception
{
    RetrieveRequest retrieveRequest = new RetrieveRequest();
    // The version in package.xml overrides the version in RetrieveRequest
    retrieveRequest.setApiVersion(API_VERSION);
    setUnpackaged(retrieveRequest);

    // Start the retrieve operation
    AsyncResult asyncResult = metadataConnection.retrieve(retrieveRequest);
    String asyncResultId = asyncResult.getId();

    // Wait for the retrieve to complete
    int poll = 0;
    long waitTimeMilliSecs = ONE_SECOND;
    RetrieveResult result = null;
    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.");
        }
```

```
            result = metadataConnection.checkRetrieveStatus(
                    asyncResultId, true);
            System.out.println("Retrieve Status: " + result.getStatus());
        } while (!result.isDone());

        if (result.getStatus() == RetrieveStatus.Failed) {
            throw new Exception(result.getErrorStatusCode() + " msg: " +
                    result.getErrorMessage());
        } else if (result.getStatus() == RetrieveStatus.Succeeded) {
            // 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.
    com.sforce.soap.metadata.Package p = parsePackage(unpackedManifest);
    request.setUnpackaged(p);
}

private com.sforce.soap.metadata.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);
            }
        }
        com.sforce.soap.metadata.Package r = new com.sforce.soap.metadata.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);
        }
    }


    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 EnterpriseConnection(loginConfig)).login(
                username, password);

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

    //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 `icon` field was added to the `CustomTab` for API version 14.0. If you retrieve components for version 13.0 or earlier, the components will not include the `icon` field.

Name	Type	Description
		Note: In API version 31.0 and later, the API version that's specified in `package.xml` is used for the `retrieve()` call and overrides the version in the `apiVersion` field. If the version is not specified in `package.xml`, the version in this field is used.
`packageNames`	string[]	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.
`singlePackage`	boolean	Specifies whether only a single package is being retrieved (`true`) or not (`false`). If `false`, then more than one package is being retrieved.
`specificFiles`	string[]	A list of file names to be retrieved. If a value is specified for this property, `packageNames` must be set to `null` and `singlePackage` must be set to `true`.
`unpackaged`	Package	A list of components to retrieve that are not in a package.

`checkRetrieveStatus()`

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

Syntax

In API version 34.0 and later:

```
RetrieveResult = metadatabinding.checkRetrieveStatus(ID id, boolean includeZip);
```

In API version 33.0 and earlier:

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

Usage

Use `checkRetrieveStatus()` to check the progress of the metadata `retrieve()` operation. The RetrieveResult object that this method returns indicates when the asynchronous `retrieve()` call is completed. If the retrieval is completed, RetrieveResult contains the zip file contents by default. Use the following process to retrieve metadata components with the `retrieve()` call.

1. Issue a `retrieve()` call to start the asynchronous retrieval. An AsyncResult object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkRetrieveStatus()` call and pass in the `id` value from the AsyncResult object from the first step. Check the value of the `done` field of the returned RetrieveResult. If it is `true`, this means that the call is completed and proceed to the next step. Otherwise, repeat this step to call `checkRetrieveStatus()` again until the `done` field is `true`.
3. Retrieve the zip file (`zipFile` field) and other desired fields from RetrieveResult that was returned by the final call to `checkRetrieveStatus()` in the previous step.

In API version 31.0 and later, the process of making a `retrieve()` call has been simplified. You no longer have to call `checkStatus()` after a `retrieve()` call to obtain the status of the retrieve operation. Instead, make calls to `checkRetrieveStatus()` only. If the retrieve operation is in progress, call `checkRetrieveStatus()` again until the retrieve operation is completed. The `checkStatus()` call is still supported in versions API version 30.0 or earlier, but is not available in API version 31.0 and later.

Retrieving the Zip File in a Second Process

By default, `checkRetrieveStatus()` returns the zip file on the last call to this operation when the retrieval is completed (`RetrieveResult.isDone() == true`) and then deletes the zip file from the server. Subsequent calls to `checkRetrieveStatus()` for the same retrieve operation can't retrieve the zip file after it has been deleted. Starting with API version 34.0, pass a boolean value for the `includeZip` argument of `checkRetrieveStatus()` to indicate whether to retrieve the zip file. The `includeZip` argument gives you the option to retrieve the file in a separate process after the retrieval operation is completed. For example, a service polls the retrieval status by calling `checkRetrieveStatus(id, false)` in a loop. This call returns the status of the retrieval operation, but doesn't retrieve the zip file. After the retrieval operation is completed, another process, such as a background file transfer service, calls `checkRetrieveStatus(id, true)` to retrieve the zip file. This last call causes the zip file to be deleted from the server.

```
// First process: Poll the retrieval but don't retrieve the zip file.
AsyncResult asyncResult = metadataConnection.retrieve(retrieveRequest);
String asyncResultId = asyncResult.getId();
// Wait for the retrieve to complete
int poll = 0;
long waitTimeMilliSecs = ONE_SECOND;
RetrieveResult result = null;
do {
    Thread.sleep(waitTimeMilliSecs);
    // Check the status but don't retrieve zip file.
    result = metadataConnection.checkRetrieveStatus(asyncResultId, false);
} while (!result.isDone());

// Second process: Retrieve the zip file.
// For example, this process can be a background file transfer service.
// Retrieve the zip file.
result = metadataConnection.checkRetrieveStatus(asyncResultId, true);
// Get the zip file from the RetrieveResult (result) variable
if (result.getStatus() == RetrieveStatus.Succeeded) {
    ByteArrayInputStream bais = new ByteArrayInputStream(result.getZipFile());
    // ...
}
```

Sample Code—Java

See the `retrieve()` sample code for sample usage of this call.

Arguments

Name	Type	Description
`id`	ID	ID obtained from an AsyncResult object returned by a `retrieve()` call or a subsequent RetrieveResult object returned by a `checkRetrieveStatus()` call.

Name	Type	Description
`includeZip`	boolean	Set to `true` to retrieve the zip file. You can retrieve the zip file only after the retrieval operation is completed. After the zip file is retrieved, it is deleted from the server. Set to `false` to check the status of the retrieval without attempting to retrieve the zip file. If set to `null`, this argument defaults to `true`, which means that the zip file is retrieved on the last call to `checkRetrieveStatus()` when the retrieval has finished. This argument is available in API version 34.0 and later.

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.

IN THIS SECTION:

createMetadata()

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

Syntax

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

Usage

Use the `createMetadata()` call to create any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if `AllOrNoneHeader` isn't used in API version 34.0 and later, this call can save a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). In API version 33.0 and earlier, the default behavior is to only save all records when there are no failures in any record in the call (equivalent to `AllOrNoneHeader=true`).

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[]	Array of one or more metadata components. Limit: 10. (For CustomMetadata only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

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 retrieve any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

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
`metadataType`	string	The metadata type of the components to read.
`fullNames`	string[]	Array of full names of the components to read. Limit: 10. (For CustomMetadata only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

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 any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if `AllOrNoneHeader` isn't used in API version 34.0 and later, this call can save a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). In API version 33.0 and earlier, the default behavior is to only save all records when there are no failures in any record in the call (equivalent to `AllOrNoneHeader=true`).

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[]	Array of one or more metadata components you wish to update. Limit: 10. (For CustomMetadata only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

SaveResult[]

upsertMetadata()

Creates or updates one or more metadata components in your organization synchronously.

Syntax

```
UpsertResult[] = metadataConnection.upsertMetadata(Metadata[] metadata);
```

Usage

Use the `upsertMetadata()` call to create or update any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

If the specified components already exist in your organization, the `upsertMetadata()` call updates them. Otherwise, `upsertMetadata()` creates these components. Components are matched by the `fullname` field. This call executes synchronously, which means that the call returns only after the operation is completed.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if `AllOrNoneHeader` isn't used in API version 34.0 and later, this call can save a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). In API version 33.0 and earlier, the default behavior is to only save all records when there are no failures in any record in the call (equivalent to `AllOrNoneHeader=true`).

Version

Available in API version 31.0 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 (not alphabetic characters) for integer fields. In your client application, follow the data formatting rules that are specified for your programming language and development tool. (Your development tool handles the appropriate mapping of data types in SOAP messages.)

String Values

The API trims any leading and trailing white space when storing values in string fields. 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 Upserting Metadata Components

Use this process to upsert metadata components.

1. Create an array of Metadata objects that correspond to the components that you want to create or update. All components must be of the same type.
2. Invoke `upsertMetadata()`, passing in the array of metadata components that you created in the previous step.

   The `upsertMetadata()` call returns an array of `UpsertResult` objects. Each returned `UpsertResult` corresponds to a component that you upserted and contains information about the upsert operation—whether the operation was successful, the name of the component that was upserted, a flag indicating whether the component was created, and any errors that were returned if the operation wasn't successful.

Sample Code—Java

```
public void upsertMetadataSample() {
    try {
        // Create custom object to upsert
        CustomObject co = new CustomObject();
        String name = "MyCustomObject";
        co.setFullName(name + "__c");
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Upserted 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("CustomField1");
        co.setNameField(nf);

        // Upsert the custom object
        UpsertResult[] results = metadataConnection
                .upsertMetadata(new Metadata[] { co });

        for (UpsertResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Success!");
                if (r.isCreated()) {
                    System.out.println("Created component: "
                            + r.getFullName());
                } else {
                    System.out.println("Updated component: "
                            + r.getFullName());
                }
            } else {
                System.out
                .println("Errors were encountered while upserting "
                        + 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[]	An array of one or more metadata components that you want to create or update Limit: 10. You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

UpsertResult[]

`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 any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if the `AllOrNoneHeader` isn't used in any API version, this call can delete a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). If `AllOrNoneHeader` is set to `true`, no records are deleted if one or more records cause a failure.

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
`metadataType`	string	The metadata type of the components to delete.
`fullNames`	string[]	Array of full names of the components to delete. Limit: 10. (For CustomMetadata only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

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

`create()`

Deprecated. Adds one or more new metadata components to your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `createMetadata()` instead.

This call can be used to create any of the objects that extend Metadata. For more details, see Metadata Components and Types on page 112.

Syntax

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

Usage

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

Version

This call is available in API version 30.0 and earlier only. This call is not available in API version 31.0 and later. Use `createMetadata()` instead.

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 112.

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 triy 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 6 for sample Java code using the `create()` call.

Arguments

Name	Type	Description
`metadata`	Metadata[]	Array of one or more metadata components. Limit: 10. 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.

Response

AsyncResult[]

SEE ALSO:

- createMetadata()
- update()
- delete()
- checkStatus()

`delete()`

Deprecated. Deletes one or more components from your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `deleteMetadata()` instead.

You can use this call to delete any of the objects that extend Metadata. For more details, see Metadata Components and Types on page 112.

Syntax

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

Usage

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

Version

This call is available in API version 30.0 and earlier only. This call is not available in API version 31.0 and later. Use `deleteMetadata()` instead.

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[]	Array of one or more metadata components. You only need to set the `fullName` field in the Metadata object. Limit: 10. 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.

Response

AsyncResult[]

SEE ALSO:

- deleteMetadata()
- create()
- update()
- checkStatus()

`update()`

Deprecated. Updates one or more components in your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `updateMetadata()` or `renameMetadata()` instead.

This call can be used to update any of the objects that extend Metadata. For more details, see Metadata Components and Types on page 112.

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.

Version

This call is available in API version 30.0 and earlier only. This call is not available in API version 31.0 and later. Use `updateMetadata()` instead to update metadata components or `renameMetadata()` to rename a metadata component.

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[]	Array of one or more UpdateMetadata data structures that represent the components you wish to update. Limit: 10. 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.

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 112. 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 `Foo__c`. 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[]

SEE ALSO:

- updateMetadata()
- create()
- delete()
- checkStatus()

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.

- (Deprecated) `checkStatus()`
- `describeMetadata()`
- `describeValueType()`
- `listMetadata()`

`checkStatus()`

Deprecated. Checks the status of asynchronous metadata calls `create()`, `update()`, or `delete()`, or the declarative metadata call `retrieve()`. This call is removed as of API version 31.0 and is available only in earlier versions.

Note: Starting in API version 29.0, you no longer have to call `checkStatus()` after a `deploy()` call to get information about deployments. Similarly, starting in API version 31.0, you no longer have to call `checkStatus()` after a `retrieve()` call. The `checkStatus()` call has been replaced by checkDeployStatus() and checkRetrieveStatus() for deploy and retrieve operations respectively.

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.

Version

This call is available only in API version 30.0 and earlier. This call is not available in API version 31.0 and later.

Sample Code—Java

See Step 3: Walk Through the Java Sample Code on page 6 for sample Java code using this call.

Arguments

Name	Type	Description
ids	ID[]	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 that 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 metadata types.

Syntax

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

Arguments

Name	Type	Description
apiVersion	double	The API version for which you want metadata; for example, 36.0.

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

When to Use `describeMetadata()` and `describeValueType()`?

Use the `describeMetadata()` call to get high-level information about all the metadata types that are available for your organization, such as type names and file suffixes. Use the `describeValueType()` call to get granular information about a specific metadata type, such as fields contained within the type.

`describeValueType()`

Retrieves the metadata describing a given metadata type (value type).

`describeValueType()` accepts a namespace and a type name, and returns a `DescribeValueTypeResult` object. This call is available in API version 33.0 and later.

Syntax

```
DescribeValueTypeResult = connection.describeValueType("{namespace}type_name");
```

Example

Describe Apex class metadata in the Metadata namespace:

```
DescribeValueTypeResult =
metadataConnection.describeValueType("{http://soap.sforce.com/2006/04/metadata}ApexClass");
```

Describe Apex class metadata in the Tooling namespace:

```
DescribeValueTypeResult =
toolingConnection.describeValueType("{urn:metadata.tooling.soap.sforce.com}ApexClass");
```

Arguments

Name	Type	Description
`type`	string	The name of the metadata type for which you want metadata; for example, `ApexClass`. Include the namespace.

Permissions

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

Sample Code—Java

The following example describes several metadata types by specifying the Metadata namespace. Each metadata type is described using the helper method, `doDescribe()`, which calls the `describeValueType()` Metadata API call. The sample retrieves information from the returned `DescribeValueTypeResult`: a property, the parent field (if any), and the fields. Next, the sample iterates through the fields and outputs information about each field.

```
public void describeValueType() throws ConnectionException {
    doDescribe("{http://soap.sforce.com/2006/04/metadata}CustomObject");
    doDescribe("{http://soap.sforce.com/2006/04/metadata}CustomField");
    doDescribe("{http://soap.sforce.com/2006/04/metadata}EmailTemplate");
}

public void doDescribe(String type) throws ConnectionException {
    DescribeValueTypeResult result = metadataConnection.describeValueType(type);
    StringBuffer sb = new StringBuffer();

    sb.append("Describing " + type + " ...\n");

    if (result.getApiCreatable() == true) {
        sb.append("Is API creatable.\n");
    } else {
        sb.append("Is not API creatable.\n");
    }

    ValueTypeField parentField = result.getParentField();
    if (parentField != null) {
        sb.append("** Parent type fields **\n");
        if (parentField.getIsForeignKey()) {
            sb.append("This field is a foreign key.\n");
            for (String fkDomain : parentField.getForeignKeyDomain()) {
                sb.append("Foreign key domain: " + fkDomain + "\n");
            }
        }
    }

    sb.append("** Value type fields **\n");
    for(ValueTypeField field : result.getValueTypeFields()) {
        sb.append("***********************************************\n");
        sb.append("Name: " + field.getName() + "\n");
```

```
        sb.append("SoapType: " + field.getSoapType() + "\n");
        if (field.getIsForeignKey()) {
            sb.append("This field is a foreign key.\n");
            for (String fkDomain : field.getForeignKeyDomain()) {
                sb.append("Foreign key domain: " + fkDomain + "\n");
            }
        }
        sb.append("************************************************\n");
    }
    System.out.println(sb.toString());
}
```

To run the previous example with the Tooling WSDL, replace the namespace with the Tooling namespace in the helper function call as follows. Also, use the Tooling connection instead of the Metadata connection to make the `describeValueType()` call.

```
    doDescribe("{urn:metadata.tooling.soap.sforce.com}CustomObject");
    doDescribe("{urn:metadata.tooling.soap.sforce.com}CustomField");
    doDescribe("{urn:metadata.tooling.soap.sforce.com}EmailTemplate");
```

After you run the sample, the output looks similar to the following.

```
Describing {http://soap.sforce.com/2006/04/metadata}CustomObject ...
Is API creatable.
** Value type fields **
************************************************
Name: actionOverrides
SoapType: ActionOverride
************************************************
************************************************
Name: allowInChatterGroups
SoapType: boolean
************************************************
************************************************
Name: articleTypeChannelDisplay
SoapType: ArticleTypeChannelDisplay
************************************************
************************************************
Name: businessProcesses
SoapType: BusinessProcess
************************************************
************************************************
Name: compactLayoutAssignment
SoapType: string
************************************************
************************************************
Name: compactLayouts
SoapType: CompactLayout
************************************************
************************************************
Name: customHelp
SoapType: string
This field is a foreign key.
Foreign key domain: ApexPage
Foreign key domain: Scontrol
```

```
************************************************
<The rest of the output for CustomObject has been omitted for brevity.>

Describing {http://soap.sforce.com/2006/04/metadata}CustomField ...
Is API creatable.
** Parent type fields **
This field is a foreign key.
Foreign key domain: CustomObject
** Value type fields **
************************************************
Name: caseSensitive
SoapType: boolean
************************************************
************************************************
Name: customDataType
SoapType: string
************************************************
************************************************
Name: defaultValue
SoapType: string
************************************************

<The rest of the output has been omitted for brevity.>
```

Response

DescribeValueTypeResult

`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 and RecordType.

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 or Layout 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 = 36.0;
    // 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, such as Dashboard, Document, EmailTemplate, or Report.
`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.

IN THIS SECTION:

AsyncResult

Contains the ID of a deployment or retrieval. In API version 28.0 and earlier, contains status information of any asynchronous metadata call.

API Version 31.0 and Later

In API version 31.0, the process of retrieving metadata has been simplified and retrieval properties have been moved to RetrieveResult. Also, the asynchronous `create()`, `update()`, and `delete()` calls have been removed. Therefore, only the `id` field in AsyncResult is used. The `id` field is the ID of a deployment or retrieval.

AsyncResult is returned by the following asynchronous calls.

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

AsyncResult has the following field that is in use.

Name	Type	Description
`id`	ID	Required. The ID of the component that's being deployed or retrieved.

All fields in AsyncResult other than `id` are deprecated as of API version 31.0. These fields exist but are no longer in use.

- `done`
- `message`
- `state`
- `statusCode`

API Versions 29.0 and 30.0

In API version 29.0, Salesforce moved several properties from the AsyncResult object to the DeployResult object and added several new ones, to improve the process for getting information about deployments. For more information about these changes, see `deploy()`.

In API versions 29.0 and 30.0, AsyncResult is returned by the same asynchronous calls as in API version 28.0 and earlier, but it has different fields.

Name	Type	Description
`done`	boolean	Required. Indicates whether the call has been completed (`true`) or not (`false`).
`id`	ID	Required. The ID of the component that's being created, updated, deleted, deployed, or retrieved.
`message`	string	The message that corresponds to the returned `statusCode` field, if any.
`state`	AsyncRequestState (enumeration of type string)	Required. The `AsyncRequestState` object has one of four possible values. • `Queued`: This call has not started. It is waiting in a queue. • `InProgress`: This call has started but has not been completed. • `Completed`: This call has been completed. • `Error`: An error occurred. See the `statusCode` for more information.

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

API Version 28.0 and Earlier

In API version 28.0 and earlier, AsyncResult is returned by the following asynchronous calls.

- `deploy()`
- `retrieve()`
- `create()`
- `update()`
- `delete()`

Use the `checkStatus()` call against each object to discover when the call is completed for that object. Salesforce updates each AsyncResult object as the call is completed or when errors occur.

Similarly, the `deploy()` and `retrieve()` calls use AsyncResult, though you must subsequently use `checkDeployStatus()` or `checkRetrieveStatus()` respectively to get more status information for the deployment or retrieval.

AsyncResult has the following fields.

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 (`true`) or not (`false`). 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 relevant only for the `deploy()` call.
`done`	boolean	Required. Indicates whether the call has been completed (`true`) or not (`false`).
`id`	ID	Required. The ID of the component that's being created, updated, deleted, deployed, or retrieved.
`message`	string	The message that corresponds to the returned `statusCode` field, 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 relevant only for the `deploy()` call.
`numberComponentsDeployed`	int	The number of components that have been deployed for this deployment. This field in conjunction with the `numberComponentsTotal` field gives you an indication of the progress of the deployment. This field is available in API version 16.0 and later and is relevant only for the `deploy()` call.

Name	Type	Description
`numberComponentsTotal`	int	The total number of components in the deployment. This field in conjunction with the `numberComponentsDeployed` field gives you an indication of the progress of the deployment. This field is available in API version 16.0 and later and is relevant only for the `deploy()` call.
`numberTestErrors`	int	The number of Apex tests that generated errors during this deployment. This field is available in API version 16.0 and later and is relevant only for the `deploy()` call.
`numberTestsCompleted`	int	The number of Apex tests that have been completed for this deployment. This field in conjunction with the `numberTestsTotal` 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 relevant only for the `deploy()` call.
`numberTestsTotal`	int	The total number of Apex tests in the deployment. This field in conjunction with the `numberTestsCompleted` 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 components that are being deployed. This field is available in API version 16.0 and later and is relevant only for the `deploy()` call.
`secondsToWait`	int	This field is no longer supported for API version 13.0 and later and is provided only for backward compatibility. The field was removed in API version 17.0. Indicates the number of seconds before the call is likel to be completed. This is an estimate only. A reasonable approach is to wait one second before calling `checkStatus()` to see if the operation is complete. Double your wait time for each successive iteration of `checkStatus()` calls until the operation is complete.
`state`	AsyncRequestState (enumeration of type string)	Required. The `AsyncRequestState` object has one of four possible values. • `Queued`: This call has not started. It is waiting in a queue. • `InProgress`: This call has started but has not been completed. • `Completed`: This call has been completed. • `Error`: An error occurred. See the `statusCode` for more information.
`stateDetail`	string	Indicates which component is being deployed or which Apex test class is running. This field is available in API version 16.0 and later and is relevant only for the `deploy()` call.
`stateDetailLastModifiedDate`	dateTime	The date and time when the `stateDetail` field was last modified. This field is available in API version 16.0 and later and is relevant only for the `deploy()` call.
`statusCode`	StatusCode (enumeration of type string)	If an error occurred during the `create()`, `update()`, `delete()`, or `deploy()` call, a status code is returned, and the message that corresponds to the status code is returned in the `message` field. For a description of each StatusCode value, see "StatusCode" in the SOAP API Developer's Guide.

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 `cancelDeploy()`, has completed (`true`) or not (`false`). When a deployment hasn't started yet and is still in the queue, the deployment is canceled immediately with the `cancelDeploy()` call and this field returns `true`. Otherwise, this field returns `false` 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, Salesforce moved several properties from the AsyncResult on page 85 object to the DeployResult object to improve the process for getting information about deployments. For more information about these changes, see `deploy()` on page 31.

For API version 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 (`true`) or not (`false`). 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.

Name	Type	Description
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 deploy() call was received.
details	DeployDetails[]	Provides the details of a deployment that is in-progress or ended, if the includeDetails parameter is set to true in the checkDeployStatus() call.
done	boolean	Indicates whether the server finished processing the deploy() call for the specified id.
errorMessage	string	Message corresponding to the values in the errorStatusCode field, if any.
errorStatusCode	string	If an error occurred during the deploy() call, a status code is returned, and the message corresponding to the status code is returned in the errorMessagefield. For a description of each StatusCode value, see "StatusCode" in the *SOAP API Developer's Guide*.
ignoreWarnings	boolean	Optional. Defaults to false. Specifies whether a deployment should continue even if the deployment generates warnings. Do not set this argument to true 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 numberComponentsTotal 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 numberComponentsDeployed 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 completedApex tests for this deployment. Use this value with the numberTestsTotal 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 numberTestsCompleted 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.

Name	Type	Description
`runTestsEnabled`	boolean	Indicates whether Apex tests were run as part of this deployment (`true`) or not (`false`). Tests are either automatically run as part of a deployment or can be set to run in `DeployOptions` for the `deploy()` 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 true. Indicates whether any failure causes a complete rollback (`true`) or not (`false`). If `false`, whatever set of actions can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to `true` if you are deploying to a production organization.
`startDate`	dateTime	Timestamp for when the deployment process began.
`stateDetail`	string	Indicates which component is 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: • `Pending` • `InProgress` • `Succeeded` • `SucceededPartial` • `Failed` • `Canceling` • `Canceled`
`success`	boolean	Indicates whether the deployment was successful (`true`) or not (`false`).

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 DeployMessage objects containing deployment errors for each component.
`componentSuccesses`	DeployMessage[]	One or more DeployMessage objects containing successful deployment details for each component.
`retrieveResult`	RetrieveResult	If the `performRetrieve` parameter was specified for the `deploy()` call, a `retrieve()` call is performed immediately after the `deploy()` process completes. This field contains the results of that retrieval.

Name	Type	Description
`runTestResult`	RunTestsResult	If tests were run for the `deploy()` call, this field contains the test results. 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 version 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 `deploy()` call.
`retrieveResult`	RetrieveResult	If the `performRetrieve` parameter was specified for the `deploy()` call, a `retrieve()` call is performed immediately after the `deploy()` process completes. This field contains the results of that retrieval.
`runTestResult`	RunTestsResult	If tests were run for the `deploy()` call, this field contains the test results.
`success`	boolean	Indicates whether the deployment was successful (`true`) or not (`false`).

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 `true`, the component was changed as a result of this deployment. If `false`, 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 `true`, the component was created as a result of this deployment. If `false`, 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 `true`, the component was deleted as a result of this deployment. If `false`, the component was either new or modified as result of the deployment.
`fileName`	string	The name of the file in the `.zip` file used to deploy this component.

Name	Type	Description
`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: • `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 (`true`) or not (`false`).

RunTestsResult

Contains information about the execution of unit tests, including whether unit tests were completed successfully, code coverage results, and failures.

A RunTestsResult object has the following properties

Name	Type	Description
`apexLogId`	string	The ID of an ApexLog object that is created at the end of a test run. The ApexLog object is created if there is an active trace flag on the user running an Apex test, or on a class or trigger being executed. This field is available in API version 35.0 and later.
`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.

Name	Type	Description
numTestsRun	int	The number of unit tests that were run.
successes	RunTestSuccess[]	An array of one or more RunTestSuccess 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.
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 or package.

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	Type	Description
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.
seeAllData	boolean	Indicates whether the test method has access to organization data (`true`) or not (`false`). This field is available in API version 33.0 and later.
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 or package.

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.
seeAllData	boolean	Indicates whether the test method has access to organization data (`true`) or not (`false`). This field is available in API version 33.0 and later.

Name	Type	Description
`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.
`partialSaveAllowed`	boolean	Indicates whether `rollbackOnError` is allowed (`true`) or not (`false`). This value is always : • `false` in production organizations. • the opposite of `testRequired`.
`testRequired`	boolean	Indicates whether tests are required (`true`) or not (`false`). This value is always the opposite of `partialSaveAllowed`.

DescribeMetadataObject

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

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

DescribeValueTypeResult

Contains information about a value type that is useful for developers working with declarative metadata.

The `describeValueType()` call returns a DescribeValueTypeResult object.

Each DescribeValueTypeResult object has the following properties.

Name	Type	Description
`apiCreatable`	boolean	Indicates whether this value type can be created through the createMetadata() call (`true`) or not (`false`). This field is available in API version 36.0 and later.
`apiDeletable`	boolean	Indicates whether this value type can be created through the deleteMetadata() call (`true`) or not (`false`). This field is available in API version 36.0 and later.
`apiReadable`	boolean	Indicates whether this value type can be created through the readMetadata() call (`true`) or not (`false`). This field is available in API version 36.0 and later.
`apiUpdatable`	boolean	Indicates whether this value type can be created through the updateMetadata() call (`true`) or not (`false`). This field is available in API version 36.0 and later.
`parentField`	ValueTypeField	Information about the parent of this value type. Parent field information is useful for metadata types that are specified with the parent in their name, such as custom fields, email templates, workflow rules, and reports. For example, the full name of a custom field includes the sObject that contains it (for example, `Account.field1__c`). Similarly, the full name of an email template includes

Name	Type	Description
		the folder where the template is stored (for example, `MyFolder/EmailTemplate1`). If the value type has no parent, this field is null. This field is available in API version 36.0 and later.
`valueTypeFields`	ValueTypeField[]	One or more metadata components and their attributes.

ValueTypeField

This object is returned as part of the DescribeValueTypeResult and represents the metadata for one field. Each ValueTypeField has the following properties.

Name	Type	Description
`fields`	ValueTypeField	The `ValueTypeField` object for the next field, if any.
`foreignKeyDomain`	string	If `isForeignKey` is `True`, `foreignKeyDomain` is the type of object, such as Account or Opportunity.
`isForeignKey`	boolean	`True` if the field is a foreign key. That means this field is the primary key in a different database table.
`isNameField`	boolean	`True` if this value type field is a `fullName` field, otherwise `False`.
`minOccurs`	int	1 if this field is required, 0 otherwise.
`name`	string	The name of this value type field. The name is null for parent fields.
`picklistValues`	PicklistEntry	The individual picklist values if the field is a picklist.
`soapType`	string	The data type of the field, such as boolean or double.
`valueRequired`	boolean	Required. Indicates whether this value type field must have a value (`true`) or can be null (`false`).

ReadResult

Contains result information for the `readMetadata` call.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
`records`	Metadata[]	An array of metadata components returned from `readMetadata()`.

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
`done`	boolean	Required. Indicates whether the `retrieve()` call is completed (`true`) or not (`false`). This field is available in API version 31.0 and later.
`errorMessage`	string	If an error occurs during the `retrieve()` call, this field contains a descriptive message about this error. This field is available in API version 31.0 and later.
`errorStatusCode`	StatusCode	If an error occurs during the `retrieve()` call, this field contains the status code for this error. This field is available in API version 31.0 and later. For a description of each StatusCode value, see "StatusCode" in the *SOAP API Developer's Guide*.
`fileProperties`	FileProperties[]	Contains information about the properties of each component in the `.zip` file, and the manifest file `package.xml`. 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 `retrieve()` call.
`status`	RetrieveStatus (enumeration of type string)	The status of the `retrieve()` call. Valid values are: • `Pending` • `InProgress` • `Succeeded` • `Failed` This field is available in API version 31.0 and later.
`success`	boolean	Indicates whether the `retrieve()` call was successful (`true`) or not (`false`). This field is available in API version 31.0 and later.
`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 `fileName` but the characters allowed are more restrictive. The `fullName` 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: • `beta` • `deleted` • `deprecated` • `installed` • `released` • `unmanaged` For more information about states of manageability for components in Force.com AppExchange packages, see "Planning the Release of Managed Packages" in the Salesforce online help.
`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.

SEE ALSO:

retrieve()

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 (`true`) or not (`false`).

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 (`true`) or not (`false`).

UpsertResult

Contains information about the result of the associated `upsertMetadata()` call.

Version

Available in API version 31.0 and later.

Properties

Name	Type	Description
`created`	boolean	Indicates whether the upsert operation resulted in the creation of the component (`true`) or not (`false`). If `false` and the upsert operation was successful, this means that the component was updated.
`errors`	Error[]	An array of errors that were returned if the operation wasn't successful.
`fullName`	string	The full name of the component that was created or updated if the operation was successful.
`success`	boolean	Indicates whether the operation was successful (`true`) or not (`false`).

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
`fields`	string[]	An array containing names of fields that affected the error condition.
`message`	string	The error message text.
`statusCode`	StatusCode	A status code corresponding to the error. For a description of each StatusCode value, see "StatusCode" in the *SOAP API Developer's Guide*.

CHAPTER 10 Metadata Types

Metadata API enables you to access some entities and feature settings that you can customize in the user interface. The following table lists the metadata types that you can retrieve or deploy and whether you can retrieve the metadata type with the wildcard character (*) in `package.xml`. For more information about using wildcards, see Working with the Zip File.

Note:

- Metadata type names are case-sensitive. Specifying a type name with an invalid case results in a deployment error.
- Metadata types don't always correspond directly to their related data types. In some cases, the information is accessible but not organized as expected. For example, dependent picklists are exposed as a type of picklist, not a separate metadata type.

Metadata Type	Allows Wildcard (*)?	Description
AccountSettings	Yes	Represents an organization's account settings for account teams, account owner report, and the **View Hierarchy** link.
ActionLinkGroupTemplate	Yes	Represents the action link group template. Action link templates let you reuse action link definitions and package and distribute action links. An action link is a button on a feed element. Clicking on an action link can take a user to another Web page, initiate a file download, or invoke an API call to an external server or Salesforce. Use action links to integrate Salesforce and third-party services into the feed. Every action link belongs to an action link group and action links within the group are mutually exclusive.
ActionOverride	No	Represents an action override on a standard or custom object. Use it to create, update, edit, or delete action overrides.
ActivitiesSettings	Yes	Represents an organization's activity settings, and its user interface settings for the calendar.
AddressSettings	Yes	Represents the configuration of country and state picklists.
AnalyticSnapshot	No	Represents a reporting snapshot. A reporting snapshot lets you report on historical data. Authorized users can save tabular or summary report results to fields on a custom object, then map those fields to corresponding fields on a target object. They can then schedule when to run the report to load the custom object's fields with the report's data. Reporting snapshots enable you to work with report data similarly to how you work with other records in Salesforce.
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.

Metadata Type	Allows Wildcard (*)?	Description
ApexComponent	Yes	Represents a Visualforce component.
ApexPage	Yes	Represents a Visualforce page.
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.
AppMenu	Yes	Represents the Force.com app menu or the Salesforce1 navigation menu.
ApprovalProcess	Yes (See description.)	Represents the metadata associated with an approval process. An approval process is an automated process your organization can use to approve records in Salesforce. An approval process specifies the steps necessary for a record to be approved and who must approve it at each step. A step can apply to all records included in the process, or just records that meet certain administrator-defined criteria. An approval process also specifies the actions to take when a record is approved, rejected, recalled, or first submitted for approval. Use the wildcard (*) symbol to retrieve all approval processes for all objects. You can't use it to retrieve a subset of approval processes; syntax such as `Lead.*` is not supported.
ArticleType	Yes	Represents the metadata associated with an article type.
AssignmentRules	Yes	Represents assignment rules that allow you to automatically route cases to the appropriate users or queues.
AuthProvider	Yes	Represents an authentication provider (or auth provider) in your organization. An auth provider enables users to log in to your Salesforce organization using their login credentials from an external service provider such as Facebook© or Janrain©.
AuraDefinitionBundle	Yes	Represents a Lightning definition bundle. A bundle contains a Lightning definition and all its related resources. The definition can be a component, application, event, interface, or a tokens collection.
AutoResponseRules	Yes	Represents an auto-response rule that sets conditions for sending automatic email responses to lead or case submissions based on the attributes of the submitted record.
BaseSharingRule	Yes	Represents the base container for criteria-based and owner-based sharing rules.
BusinessHoursSettings	Yes	Represents the metadata used to manage settings for business hours and holidays in entitlements, entitlement templates, campaigns, and cases.
BusinessProcess	(See description.)	The BusinessProcess metadata type enables you to display different picklist values for users based on their profile. Supports wildcard (*) only if a RecordType is specified.

Metadata Type	Allows Wildcard (*)?	Description
CallCenter	Yes	Represents the Call Center definition used to integrate Salesforce with a third-party computer-telephony integration (CTI) system.
CaseSettings	Yes	Represents an organization's case settings, such as the default case owner, which case-related features are enabled, and which email templates are used for various case activities.
Certificate	Yes	Represents a certificate used for digital signatures which verify that requests are coming from your org. Certificates are used for either authenticated single sign-on with an external website, or when using your org as an identity provider.
ChatterAnswersSettings	Yes	Represents the metadata used to manage settings for Chatter Answers.
CompanySettings	Yes	Represents global settings that affect multiple features in your organization.
Community (Zone)	Yes	Represents a zone that contains Ideas or Chatter Answers objects. Zones are shared by the Ideas, Answers, and Chatter Answers features, allowing you to view and create zones from those locations.
CompactLayout	Yes	Represents the metadata associated with a compact layout.
ConnectedApp	Yes	Represents a connected app configuration. A connected app integrates an application with Salesforce using APIs. Connected apps use standard SAML and OAuth protocols to authenticate, provide Single Sign-On, and provide tokens for use with Salesforce APIs. In addition to standard OAuth capabilities, connected apps allow administrators to set various security policies and have explicit control over who may use the corresponding applications.
ContractSettings	Yes	Represents contract settings.
CorsWhitelistOrigin	Yes	Represents an origin in the CORS whitelist.
CriteriaBasedSharingRule	Yes	Represents a criteria-based sharing rule. CriteriaBasedSharingRule enables you to share records based on specific criteria. It contains metadata for the following criteria-based sharing rules: Accounts, Campaigns, Cases, Contacts, Custom Objects, Leads, and Opportunities.
CustomApplication	Yes	CustomApplication represents a custom or standard application. In API version 29.0 and earlier, CustomApplication represents only a custom application. An application is a list of tab references, with a description and a logo.
CustomApplicationComponent	Yes	Represents a custom console component (Visualforce page) assigned to a CustomApplication that is marked as a Salesforce console. Custom console components extend the capabilities of Salesforce console apps.
CustomFeedFilter	Yes	Represents a custom feed filter that limits the feed view to feeds from the Cases object. The custom feed filter shows only feed items that satisfy the criteria specified in the CustomFeedFilter definition.

Metadata Type	Allows Wildcard (*)?	Description
CustomField	No	Represents the metadata associated with a field. Use this metadata type to create, update, or delete custom field definitions on standard, custom, and external objects or standard field definitions on standard objects.
CustomLabel	No	Represents a custom label that can be localized and used in different languages, countries, and currencies. Use this type instead of CustomLabels if you want to retrieve custom labels by name.
Custom Metadata Types (CustomObject)	Yes	Represents the metadata associated with a custom metadata type.
CustomMetadata	Yes	Represents a record of a custom metadata type.
CustomLabels	Yes	This metadata type allows you to create custom labels that can be localized for use in different languages, countries, and currencies.
CustomObject	(See description.)	Represents a custom object that stores data unique to your organization or an external object that maps to data stored outside Salesforce. You can also use this metadata type to work with customizations of standard objects, such as Accounts. It contains the following types: Action Overrides, Business Processes, Custom Fields, Field Sets, List Views, Named Filters (Lookup Filters), Picklists (including Dependent Picklists), Record Types, Search Layouts, Sharing Reasons, Sharing Recalculations, Validation Rules, Weblinks, and Field Types. Supports wildcard (*) for Field Sets and Record Types, but not for other components.
CustomObjectTranslation	Yes	This metadata type allows you to translate custom objects for a variety of languages.
CustomPageWebLink	Yes	Represents a web link defined in a home page component.
CustomPermission	Yes	Represents a permission that grants access to a custom feature.
CustomSite	Yes	Represents a Force.com site. Force.com Sites enables you to create public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password.
CustomTab	Yes	Represents a custom tab. Display custom object data or other Web content using custom tabs in Salesforce. When a tab displays a custom object, the tab name is the same as the custom object name; for page, s-control, or URL tabs, the name is arbitrary.
Dashboard	No	Represents a dashboard. Dashboards are visual representations of data that allow you to see key metrics and performance at a glance.
DataCategoryGroup	Yes	Represents a data category group.
DelegateGroup	Yes	Represents a group of users who have the same administrative privileges. These groups are different from public groups used for sharing.

Metadata Type	Allows Wildcard (*)?	Description
Document	No	Represents a Document. All documents must be in a document folder, for example `sampleFolder/TestDocument`.
EmailTemplate	No	Represents an email template.
EntitlementProcess	Yes	Represents the settings for an entitlement process.
EntitlementSettings	Yes	Represents an organization's entitlement settings.
EntitlementTemplate	Yes	Represents an entitlement template. Entitlement templates are predefined terms of customer support that you can quickly add to products. For example, you can create entitlement templates for Web or phone support so that users can easily add entitlements to products offered to customers.
EventDelivery	Yes	Reserved for future use.
EventSubscription	Yes	Reserved for future use.
EventType	Yes	Reserved for future use.
ExternalDataSource	Yes	Represents the metadata associated with an external data source. Create external data sources to manage connection details for integration with data and content that are stored outside your Salesforce org.
FieldSet	Yes	Represents a field set. A field set is a grouping of fields. For example, you could have a field set that contains fields describing a user's first name, middle name, last name, and business title.
FlexiPage	Yes	Represents the metadata associated with a Lightning Page. A Lightning Page is the home page for an app that appears as a menu item in the Salesforce1 navigation menu.
Flow	Yes	Represents the metadata associated with a flow. With Flow, you can create an application that navigates users through a series of screens to query and update records in the database. You can also execute logic and provide branching capability based on user input to build dynamic applications.
Folder	No	Represents a folder. A folder can contain documents, email templates, reports, or dashboards. You must specify the folder type (Document, EmailTemplate, Report, or Dashboard) to retrieve or deploy.
FolderShare	No	Represents the settings for enhanced analytics folder sharing. Users can control access to reports or dashboards by giving others Viewer, Editor or Manager access to the folder that contains the report or dashboard.
ForecastingSettings	Yes	Represents the Collaborative Forecasts settings options.
Group	Yes	Represents a set of public groups, which can have users, roles, and other groups.
HomePageComponent	Yes	Represents the metadata associated with a home page component. You can customize the Home tab to include components such as sidebar links, a company logo, a dashboard snapshot, or custom components that you create.

Metadata Type	Allows Wildcard (*)?	Description
HomePageLayout	Yes	Represents the metadata associated with a home page layout. You can customize home page layouts and assign the layouts to users based on their user profile.
IdeasSettings	Yes	Represents the metadata used to manage settings for Ideas.
InstalledPackage	Yes	Represents a package to be installed or uninstalled. Deploying a newer version of a currently installed package upgrades the package.
KeywordList	Yes	Represents a list of keywords used in community moderation. This keyword list is a type of moderation criteria that defines offensive language or inappropriate content that you don't want in your community.
KnowledgeSettings	Yes	Represents the metadata used to manage settings for Salesforce Knowledge.
Layout	Yes	Represents the metadata associated with a page layout.
Letterhead	No	Represents formatting options for the letterhead in an email template. Letterheads define the look and feel of your HTML email templates. Your HTML email templates can inherit the logo, color, and text settings from a letterhead.
ListView	No	ListView allows you to see a filtered list of records, such as contacts, accounts, or custom objects.
LiveAgentSettings	Yes	Represents an organization's Live Agent settings, such as whether or not Live Agent is enabled.
LiveChatAgentConfig	Yes	Represents the configuration of an organization's Live Agent deployment, such as how many chats can be assigned to an agent and whether or not chat sounds are enabled.
LiveChatButton	Yes	Represents a Live Agent deployment's settings for the button that customers click to chat with an agent and the chat window, such as the label that appears on the button and the pre-chat form that appears before a live chat begins.
LiveChatDeployment	Yes	Represents the configuration settings for a specific Live Agent deployment, such as the branding image for the deployment and whether or not chat transcripts are automatically saved.
LiveChatSensitiveDataRule	Yes	Represents a rule for masking or deleting data of a specified pattern. Written as a regular expression (regex). This object is available in API version 35.0 and later.
ManagedTopics	Yes	Represents navigational and featured topics managed in a community.
MatchingRule	Yes	Represents a matching rule that is used to identify duplicate records.
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.

Metadata Type	Allows Wildcard (*)?	Description
MilestoneType	Yes	Represents the name and description of a milestone, which you can use in an entitlement process to track important steps in cases.
MobileSettings	Yes	Represents an organization's mobile settings, such as mobile Chatter settings, whether Mobile Lite is enabled for an organization, and so on.
ModerationRule	Yes	Represents a rule used in your community to moderate user-generated content. Each rule specifies the user-generated content the rule applies to, the criteria to enforce the rule on, and the moderation action to take. You can create rules that block, flag, or replace user-generated content that contains offensive language or inappropriate content.
NamedCredential	Yes	Represents a named credential, which specifies the URL of a callout endpoint and its required authentication parameters in one definition. A named credential can be specified as an endpoint to simplify the setup of authenticated callouts.
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 `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.
NameSettings	Yes	Enables or disables middle name and suffix attributes for the following person objects: Contact, Lead, Person Account, and User.
Network	Yes	Represents a community. Communities are branded spaces for your employees, customers, and partners to connect. You can customize and create communities to meet your business needs, then transition seamlessly between them. Use the Network component for Salesforce Communities. If you want to create zones that contain Chatter Answers and Ideas, use the Community (Zone) component.
OpportunitySettings	No	Represents organization preferences for features such as automatic opportunity updates and similar-opportunity filters.
OrderSettings	Yes	Represents order settings.
OwnerSharingRule	Yes	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 contains metadata for the following specific owner-sharing rules: Accounts, Campaigns, Cases, Contacts, Custom Objects, Leads, Account Territory and Opportunities.
Package	No	Used to specify metadata components to be retrieved as part of a `retrieve()` call, or to define a package of components.
PathAssistant	Yes	Represents Sales Path records for Opportunity.
PathAssistantSettings	No	Represents the Sales Path preference setting.

Metadata Type	Allows Wildcard (*)?	Description
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, such as an account.
PlatformCachePartition	Yes	Represents a partition in the Platform Cache.
Portal	Yes	The Portal metadata type represents a partner portal or Customer Portal.
PostTemplate	Yes	Represents the metadata associated with an approval post template for Approvals in Chatter. With approval post templates, you can customize the information included in approval request posts that appear in Chatter feeds.
ProductSettings	No	Represents organization preferences for quantity schedules, revenue schedules, and active flag interaction with prices.
Profile	Yes	Represents a user profile. A profile defines a user's permission to perform different functions within Salesforce.
Queue	Yes	Represents a holding area for items before they are processed.
QuickAction	Yes	Represents a specified create or update quick 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.
QuoteSettings	No	Enables or disables Quotes, which show proposed prices for products and services.
RecordType	No	Represents the metadata associated with a record type. Record types let you offer different business processes, picklist values, and page layouts to different users.
RemoteSiteSetting	Yes	Represents a remote site setting. Before any Visualforce page, Apex callout, or JavaScript code using XmlHttpRequest in an s-control or custom button can call an external site, that site must be registered in the Remote Site Settings page, or the call will fail.
Report	No	Represents a custom report.
ReportType	Yes	Represents the metadata associated with a custom report type. Custom report types allow you to build a framework from which users can create and customize reports.
Role	Yes	Represents a role in your organization.
SamlSsoConfig	Yes	Represents a SAML Single Sign-On configuration.

Metadata Type	Allows Wildcard (*)?	Description
Scontrol	Yes	Deprecated. Represents an Scontrol component, corresponding to an s-control in the Salesforce user interface.
SearchLayouts	No	Represents the metadata associated with the Search Layouts for an object. You can customize which fields to display for users in search results, search filter fields, lookup dialogs, and recent record lists on tab home pages.
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.
SharingBaseRule	No	Represents sharing rule settings such as access level and to whom access is granted.
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	Yes	Represents the base container for sharing rules, which can be criteria-based, ownership-based, or territory-based. SharingRules enables you to share records with a set of users, using rules that specify the access level for the target user group.
SharingSet	Yes	Represents a sharing set. A sharing set defines an access mapping that grants portal or community users access to objects that are associated with their accounts or contacts.
SiteDotCom	Yes	Represents a site for deployment. It extends the MetadataWithContent type and inherits its `fullName` and `content` fields.
Skill	Yes	Represents the settings for a skill used to route chats to agents in Live Agent, such as the name of the skill and which agents the skills are assigned to.
StaticResource	Yes	Represents a static resource file, often a code library in a ZIP file.
SynonymDictionary	Yes	Represents a set of synonym groups, which are groups of words or phrases that are treated as equivalent in users' searches. You can define synonym groups to optimize search results for acronyms, variations of product names, and other terminology unique to your organization.
Territory	Yes	Represents a territory in your organization.
Territory2	Yes	Represents the metadata associated with a sales territory in Territory Management 2.0.
Territory2Model	Yes	Represents the metadata associated with a territory model in Territory Management 2.0.
Territory2Rule	Yes	Represents the metadata associated with a territory assignment rule associated with an object, such as Account, in Territory Management 2.0.

Metadata Type	Allows Wildcard (*)?	Description
Territory2Settings	No	Represents the metadata for the default settings for Territory Management 2.0 users to access and modify records associated with sales territories. The standard record access settings apply to accounts and opportunities. If your organization uses *Private* default internal access for contacts or cases, you can also set access for those records.
Territory2Type	Yes	Represents the metadata for a category of territories in Territory Management 2.0. Every Territory2 must have a Territory2Type.
TransactionSecurityPolicy	Yes	Represents a transaction security policy definition.
Translations	Yes	This metadata type allows you to work with translations for various supported languages.
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 `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.
WaveTemplateBundle	Yes	Represents a Wave Analytics template bundle, which can be used to create Wave apps. A bundle contains a Wave template definition and all its related resources.
WebLink	No	Represents a WebLink defined in a custom object.
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. See `sortBy` for an example of an enumeration field of type string. 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>
```

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:

- CRUD calls, such as createMetadata() and deleteMetadata()
- File-based calls, such as deploy() and retrieve()
- Utility calls, such as listMetadata() and describeMetadata()

Unsupported Metadata Types

Some things you can customize in a Salesforce 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:

- Account Teams
- Activity Button Overrides
- Analytic Settings
- Auto-number on Customizable Standard Fields

- Campaign Influences
- Case Contact Roles
- Case Feed Layouts
- Case Team Roles
- Console Layouts
- Currency Exchange Rates
- Data Category Visibility Settings
- Delegated Administration
- Divisions
- Email Services
- Fiscal Year
- HTML Document and Attachment Settings
- Lead Settings
- Mail Merge Templates
- Mobile Administration
- Mobile Users and Devices
- Multiline layout fields for opportunity teams
- Offline Briefcase Configurations
- Opportunity Big Deal Alerts
- Opportunity Update Reminders
- Organization Wide Email Addresses
- Partner Management
- The following standard picklists: Lead.CampaignMemberStatus, Opportunity.ForecastCategoryName, and Order.Status. (All other standard picklists are supported.)
- Predefined Case Teams
- Product Schedule Setup
- Public and Resource Calendars
- Quote Templates
- Salesforce to Salesforce
- Standard fields that aren't customizable, such as autonumber fields or system fields
- Search Settings
- Self-Service Portal Font and Colors
- Self-Service Portal Settings
- Self-Service Portal Users
- Self-Service Public Solutions
- Self-Service Web-to-Case
- Site.com
- Social Account/Contact Settings
- Solution Categories
- Solution Settings

- Tag Settings
- Territory Assignment Rules
- User Interface Settings (except calendar features, which are supported in ActivitiesSettings on page 506)
- Web Links on Person Account Page Layouts
- Web-to-Lead

Special Behavior in Metadata API Deployments

Important considerations for specific types and contents of a deployment.

When deploying changes to a Salesforce org, consider how individual components in your deployment behave so you're including all the necessary changes. Use the following information to determine what to include in your deployment, and how the changes appear in the destination org.

Apex Classes and Apex Triggers

By default, changes to Apex code that have Apex jobs pending or in progress can't be deployed. To deploy these changes, do one of the following.

- Cancel Apex jobs before deploying changes to Apex code. Reschedule the jobs after the deployment.
- Enable deployments with Apex jobs in the Salesforce user interface in the Deployment Settings page.

Approval Processes

- To use approval processes on Salesforce Knowledge articles with the Metadata API, the article type must be deployed. For article version (_kav) in approval processes, the supported action types are: Knowledge Action, Email Alert, Field Update, and Outbound Message.
- If the approval process references any post templates that contain custom fields, then you need to resave those post templates in the originating organization before adding them to the change set. From Setup, enter *Post Templates* in the Quick Find box, then select **Post Templates**. For each post template, click **Edit** and then **Save**.
- The metadata doesn't include the order of active approval processes. You may need to reorder the approval processes in the destination organization after deployment.
- If you change the `Unique Name` of an approval process that was previously included in a change set and deployed in another organization, and you resend the approval process via a change set, a new approval process will be created upon deployment in the other organization. The previously deployed approval process will not be modified.

Custom Fields

Starting in API version 30.0, when deploying a new custom field, the default values for the `editable` and `readable` fields in profile field permissions are `false`. To override the default values, include field permissions for the new field in your profiles.

Custom Objects

Using API version 29.0, you can't change the `sharingModel` of an object using the Metadata API. You must manually make this change to the target org through the user interface.

Starting with API version 30.0, you can change the `sharingModel` of an object for internal users using the Metadata API and the user interface.

Connected App

- You cannot set the `consumerKey` in the Metadata API. It is included in a retrieve operation for informational purposes. If you try to move the connected app to another org, you must remove the `consumerKey` from the .zip file before the deployment to an org. A new key will be generated in the destination org.
- Mobile settings of connected apps are not supported in change sets and must be manually migrated.

Named Credentials

The following callout options for named credentials can be set only via the user interface. If the default values aren't appropriate in the destination organization, the admin for that organization must manually configure the named credential after deployment.

- `Generate Authorization Header`—Default: Enabled
- `Allow Merge Fields in HTTP Header`—Default: Disabled
- `Allow Merge Fields in HTTP Body`—Default: Disabled

Page Layout

A deployment containing page layout assignments replaces all existing page layout assignments in the destination org with those specified in the .zip file. Existing page layouts in the org disappear if they're not included in the .zip file. Always include all page layouts for all required record types in the .zip file.

Profiles

If a package includes a profile with a name that doesn't exist in the target org, a new profile is created with that name. If the deployed profile doesn't specify any permissions or settings, the resulting profile consists of all the permissions and settings in the Standard Profile.

Sharing

- Simultaneously updating the `sharingModel` field for an object and adding a new sharing rule isn't supported in the Metadata API, regardless of which object you're updating. For example, you can add a sharing rule when the org-wide default is public, and subsequently update the `sharingModel`. This would result in a single sharing recalculation.
- You might encounter an error if you're deploying a change set with a custom object that has a parent-child relationship without the master/detail field in the same change set. To resolve this error, include the master/detail custom field in the change set, even if you haven't changed the org-wide default.

Workflow

Test mode for flow triggers isn't supported in the Metadata API. If you want a flow trigger to run the latest flow version when an administrator causes the workflow rule to fire, enable test mode via the user interface after deployment.

ActionLinkGroupTemplate

Represents the action link group template. Action link templates let you reuse action link definitions and package and distribute action links. An action link is a button on a feed element. Clicking on an action link can take a user to another Web page, initiate a file download, or invoke an API call to an external server or Salesforce. Use action links to integrate Salesforce and third-party services into the feed. Every action link belongs to an action link group and action links within the group are mutually exclusive. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ActionLinkGroupTemplate components have the suffix `.actionLinkGroupTemplate` and are stored in the `actionLinkGroupTemplates` folder.

Version

ActionLinkGroupTemplate components are available in API version 33.0 and later.

Fields

Field Name	Field Type	Description
`actionLinkTemplates`	ActionLinkTemplate on page 117[]	Action link templates that are associated with the action link group template.
`category`	PlatformAction GroupCategory (enumeration of type string)	Required. The location of the action link group within the feed element. Values are: • `Primary`—The action link group is displayed in the body of the feed element. • `Overflow`—The action link group is displayed in the overflow menu of the feed element.
`executionsAllowed`	ActionLink ExecutionsAllowed (enumeration of type string)	Required. The number of times an action link can be executed. Values are: • `Once`—An action link can be executed only once across all users. • `OncePerUser`—An action link can be executed only once for each user. • `Unlimited`—An action link can be executed an unlimited number of times by each user. If the action link's `actionType` is `Api` or `ApiAsync`, you can't use this value.
`hoursUntilExpiration`	int	Required. The number of hours from when the action link group is created until it's removed from associated feed elements and can no longer be executed. The maximum value is 8,760.
`isPublished`	boolean	Required. If `true`, the action link group template is published. Action link group templates shouldn't be published until at least one action link template is associated with it.
`name`	string	Required. The name of the action link group template to use in code.

ActionLinkTemplate

ActionLinkTemplate components are used to create multiple action links that share properties.

Field Name	Field Type	Description
`actionUrl`	string	Required. The action link URL. For example, a `Ui` action link URL is a Web page. A `Download` action link URL is a link to the file to download. `Ui` and `Download` action link URLs are provided to clients. An `Api` or `ApiAsync` action link URL is a REST resource. `Api` and `ApiAsync` action link URLs aren't provided to clients. Links to Salesforce can be relative. All other links must be absolute and start with `https://`.
`headers`	string	Template for the HTTP headers sent when corresponding action links are invoked. This field can be used only for `Api` and `ApiAsync` action links.

Field Name	Field Type	Description
		This field can contain context variables and binding variables in the form `{!Bindings.key}`.
`isConfirmationRequired`	boolean	Required. If `true`, a confirmation dialog appears before the action is executed.
`isGroupDefault`	boolean	Required. If `true`, action links derived from this template are the default or primary action in their action groups. There can be only one default action per action group.
`label`	string	A custom label to display on the action link button. If none of the `LabelKey` values make sense for an action link, use a custom label. Set the `LabelKey` field to `None` and enter a label name in the `Label` field.
`labelKey`	string	Required. Key for the set of labels to display for these action link states: new, pending, success, failed. For example, the Approve set contains these labels: Approve, Pending, Approved, Failed. For a complete list of keys and labels, see Action Links Labels in the *Chatter REST API Developer Guide* or the *Apex Developer Guide*.
`linkType`	ActionLinkType (enumeration of type string)	Required. The type of action link. One of these values: • `Api`—The action link calls a synchronous API at the action URL. Salesforce sets the status to `SuccessfulStatus` or `FailedStatus` based on the HTTP status code returned by your server. • `ApiAsync`—The action link calls an asynchronous API at the action URL. The action remains in a `PendingStatus` state until a third party makes a request to `/connect/action-links/actionLinkId` to set the status to `SuccessfulStatus` or `FailedStatus` when the asynchronous operation is complete. • `Download`—The action link downloads a file from the action URL. • `Ui`—The action link takes the user to a Web page at the action URL.
`method`	ActionLink HttpMethod (enumeration of type string)	Required. HTTP method for the action URL. One of these values: • `HttpDelete`—Returns HTTP 204 on success. Response body or output class is empty. • `HttpGet`—Returns HTTP 200 on success. • `HttpHead`—Returns HTTP 200 on success. Response body or output class is empty. • `HttpPatch`—Returns HTTP 200 on success or HTTP 204 if the response body or output class is empty. • `HttpPost`—Returns HTTP 201 on success or HTTP 204 if the response body or output class is empty. Exceptions are the batch posting resources and methods, which return HTTP 200 on success. • `HttpPut`—Return HTTP 200 on success or HTTP 204 if the response body or output class is empty. `Ui` and `Download` action links must use `HttpGet`.

Field Name	Field Type	Description
`position`	int	Required. An integer specifying the position of the action link template relative to other action links in the group. 0 is the first position.
`requestBody`	string	Template for the HTTP request body sent when corresponding action links are invoked. This field can be used only for `Api` and `ApiAsync` action links. This field can contain context variables and binding variables in the form `{!Bindings.key}`.
`userAlias`	string	If you selected `CustomUser` or `CustomExcludedUser` for `UserVisibility`, this field is the alias for the custom user. Use the alias in a template binding to specify the custom user when an action link group is created using the template.
`userVisibility`	ActionLink UserVisibility (enumeration of type string)	Required. Who can see the action link. This value is set per action link, not per action link group. Values are: • `Creator`—Only the creator of the action link can see the action link. • `Everyone`—Everyone can see the action link. • `EveryoneButCreator`—Everyone but the creator of the action link can see the action link. • `Manager`—Only the manager of the creator of the action link can see the action link. • `CustomUser`—Only the custom user can see the action link. • `CustomExcludedUser`—Everyone but the custom user can see the action link.

Declarative Metadata Sample Definition

The following is an example of an ActionLinkGroupTemplate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ActionLinkGroupTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
   <actionLinkTemplates>
      <actionUrl>/services/data/{!Bindings.word}/chatter/feed-elements</actionUrl>
      <headers>Content-Type:{!Bindings.word3}</headers>
      <isConfirmationRequired>true</isConfirmationRequired>
      <isGroupDefault>true</isGroupDefault>
      <labelKey>Add</labelKey>
      <linkType>API</linkType>
      <method>httpPost</method>
      <position>0</position>
      <requestBody>{"body":{"messageSegments":[{"type": "Text",
      "text": "{!Bindings.word1}"}]},"subjectId": "{!Bindings.word2}",
      "feedElementType": "feedItem"}</requestBody>
      <userAlias>customExcludedUser</userAlias>
      <userVisibility>CustomExcludedUser</userVisibility>
   </actionLinkTemplates>
   <category>Primary</category>
   <executionsAllowed>OncePerUser</executionsAllowed>
```

```
    <hoursUntilExpiration>10</hoursUntilExpiration>
    <isPublished>true</isPublished>
    <name>MyPackage</name>
</ActionLinkGroupTemplate>
```

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>*</members>
        <name>ActionLinkGroupTemplate</name>
    </types>
    <version>33.0</version>
</Package>
```

Usage

If you modify action link group templates, you overwrite the related action link templates.

If you delete a published action link group template, you delete all related action link information which includes deleting all action links that were instantiated using the template from feed items.

AnalyticSnapshot

Represents a reporting snapshot. A reporting snapshot lets you report on historical data. Authorized users can save tabular or summary report results to fields on a custom object, then map those fields to corresponding fields on a target object. They can then schedule when to run the report to load the custom object's fields with the report's data. Reporting snapshots enable you to work with report data similarly to how you work with other records in Salesforce.

Declarative Metadata File Suffix and Directory Location

Force.com AnalyticSnapshot components are stored in the `analyticSnapshots` directory of the corresponding package directory. The file name matches the unique name of the reporting snapshot, and the extension is `.analyticsnapshot`.

Version

Force.com AnalyticSnapshot components are available in API version 16.0 and later.

Fields

Field	Field Type	Description
`description`	string	A description of the reporting snapshot.
`fullName`	string	The reporting snapshot name used for API access. The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an

Field	Field Type	Description
		underscore or contain two consecutive underscore characters. This field is inherited from the Metadata component.
groupColumn	string	A column that specifies which level to extract data from the source report. It is only applicable for summary reports.
mappings	AnalyticSnapshotMapping[]	A list of reporting snapshot mappings. For valid values, see AnalyticSnapshotMapping.
name	string	Required. The display name of the reporting snapshot.
runningUser	string	The username of the user whose role and *sharing* settings are used to run the reporting snapshot.
sourceReport	string	Required. The report where data will be extracted from.
targetObject	string	Required. The custom object where data will be inserted into.

AnalyticSnapshotMapping

AnalyticSnapshotMapping defines the mapping for the reporting snapshot. Valid values are:

Field	Field Type	Description
aggregateType	ReportSummaryType[] (enumeration of type string)	List that defines if and how each report field is summarized. For valid values, see ReportSummaryType.
sourceField	string	The sourceField can be one of the following: • The field on the sourceReport that you want to map to the targetField in the targetObject • A summary of a filed on the sourceReport (for Summary reports only) • A field on the reporting snapshot, such as JobName, RunningUser, or ExecutionTime (set through the user interface) **Note:** The sourceField must correspond to the sourceType you specify.
sourceType	ReportJobSourceTypes[] (enumeration of type string)	List that defines the report format for the reporting snapshot. For valid values, see ReportJobSourceTypes.
targetField	string	A field on the targetObject into which this particular sourceField will be inserted.

ReportJobSourceTypes

An enumeration of type string that defines the report format for the reporting snapshot. Valid values are:

Enumeration Value	Description
snapshot	Use this option if the sourceField contains snapshot-specific information such as JobName, RunningUser, or ExecutionTime.

Enumeration Value	Description
summary	Use this option if referencing a summary (Sum, Average, Minimum, Maximum) of a field from the sourceReport.
tabular	Use this option if referencing an available column from the sourceReport.

Declarative Metadata Sample Definition

A sample XML definition of a reporting snapshot is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<AnalyticSnapshot xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>my description</description>
    <groupColumn>INDUSTRY</groupColumn>
    <mappings>
        <aggregateType>Average</aggregateType>
        <sourceField>SALES</sourceField>
        <sourceType>summary</sourceType>
        <targetField> myObject __c.Name</targetField>
    </mappings>
    <mappings>
        <sourceField>ExecutionTime</sourceField>
        <sourceType>snapshot</sourceType>
        <targetField> myObject __c.field3__c</targetField>
    </mappings>
    <mappings>
        <sourceField>INDUSTRY</sourceField>
        <sourceType>tabular</sourceType>
        <targetField>testObject__c.Name</targetField>
    </mappings>
    <name>my snapshot</name >
    <runningUser>user@salesforce.com</runningUser>
    <sourceReport>myFolder/mytSummaryReport</sourceReport>
    <targetObject>myObject__c</targetObject>
</AnalyticSnapshot>
```

SEE ALSO:

Report

ArticleType

Represents the metadata associated with an article type. All articles in Salesforce Knowledge are assigned to an *article type*. An article's type determines the type of content it contains, its appearance, and which users can access it. For example, a simple FAQ article type might have two custom fields, `Question` and `Answer`, where article managers enter data when creating or updating FAQ articles. A more complex article type may require dozens of fields organized into several sections. Using layouts and templates, administrators can structure the article type in the most effective way for its particular content. User access to article types is controlled by permissions. For each article type, an administrator can grant "Create," "Read," "Edit," or "Delete" permissions to users. For example, the article manager might want to allow internal users to read, create, and edit FAQ article types, but let partner users only read FAQs. See "Managing Article Types" in the Salesforce online help and "Articles" in the *SOAP API Developer's Guide*.

Declarative Metadata File Suffix and Directory Location

An ArticleType is defined as a custom object and is stored in the `objects` folder. ArticleTypes have a suffix `__kav` (instead of `__c` for custom objects). ArticleType field names have a suffix of `__c` like other custom objects, and must be dot-qualified with the name of the article type to which they belong. This is shown in the following sample `package.xml` file:

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

    <types>
        <members>newarticle__kav.description__c</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>newarticle__kav</members>
        <name>CustomObject</name>
    </types>

</Package>
```

Version

ArticleTypes are available in API version 19.0 and later.

Fields

Field Name	Field Type	Description
`articleTypeChannelDisplay`	articleTypeChannelDisplay	Represents the article-type templates used to display an article in the various channels. See "Assigning Article-Type Templates" in the Salesforce online help.
`deploymentStatus`	DeploymentStatus (enumeration of type string)	A string which represents the deployment status of a custom object or field. Valid values are: • `InDevelopment` • `Deployed`
`description`	string	A description of the article type. Maximum of 1000 characters.
`fields`	CustomField[]	Represents one or more fields in the article type.
`gender`	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender.
`label`	string	Label that represents the object throughout the Salesforce user interface.
`pluralLabel`	string	Plural version of the `label` value.

Field Name	Field Type	Description
`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.

articleTypeChannelDisplay

Determines the article-type templates that are used to display an article in its channels. Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
`articleTypeTemplates`	articleTypeTemplates	Indicates which article-type template applies in the specified channel.

articleTypeTemplates

Sets the article-type template for a specific channel. If not specified, the default article-type template applies.

Field Name	Field Type	Description
`channel`	string	Specifies the channel where the article-type template applies: • `AllChannels`: all the available channels. • `App`: the Articles tab in Salesforce Knowledge. • `Pkb`: the public knowledge base. • `Csp`: the Customer Portal. • `Prm`: the partner portal. For more information about channels, see "Salesforce Knowledge Terminology" in the Salesforce online help.
`page`	string	Represents the name of the custom Visualforce page used as a custom article-type template. Use this field when you select `Page` in the template field.
`template`	string	Indicates the article-type template used for the specified channel: • `Page`: custom Visualforce page. When specifying this value, you must also set the `page` field with the Visualforce page name. • `Tab`: display the sections you defined in the layout as tabs. • `Toc`: display the sections you defined in the layout as table of content.

Declarative Metadata Sample Definitions

A sample article type definition follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <articleTypeChannelDisplay>
        <articleTypeTemplates>
            <channel>App</channel>
            <template>Tab</template>
        </articleTypeTemplates>
        <articleTypeTemplates>
            <channel>Prm</channel>
            <template>Tab</template>
        </articleTypeTemplates>
        <articleTypeTemplates>
            <channel>Csp</channel>
            <template>Tab</template>
        </articleTypeTemplates>
        <articleTypeTemplates>
            <channel>Pkb</channel>
            <template>Toc</template>
        </articleTypeTemplates>
    </articleTypeChannelDisplay>
    <deploymentStatus>Deployed</deploymentStatus>
    <description>Article type with custom fields</description>
    <fields>
        <fullName>description__c</fullName>
        <label>Description</label>
        <length>48</length>
        <type>Text</type>
    </fields>
    <label>newarticle</label>
    <pluralLabel>newarticles</pluralLabel>
</CustomObject>
```

SEE ALSO:

ArticleType Layout

ArticleType CustomField

ArticleType Layout

Represents the metadata associated with an article type page layout. Article type layouts determine which fields users can view and edit when entering data for an article, they also determine which sections appear when users view articles. The format of the article, for example whether layout sections display as subtabs or as a single page with links, is defined by the article-type template. Each article type has only one layout, but you can choose a different template for each of the article type's four channels. For more information, see "Managing Article Types" in the Salesforce online help and “Articles” in the *SOAP API Developer's Guide*

File Suffix and Directory Location

ArticleType layouts are stored in the `layouts` directory of the corresponding package directory. The prefix must match with the article type API name. The extension is `.layout`.

Version

ArticleType layouts are available in API version 19.0 and later.

Fields

Field Name	Field Type	Description
`layoutSections`	LayoutSection[]	The main sections of the layout containing the article fields. The order here determines the layout order.

LayoutSection

LayoutSection represents a section of an ArticleType layout.

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.
`label`	string	The label; either standard or custom, based on the `customLabel` flag.
`layoutColumns`	LayoutColumn[]	The columns of the layout, depending on the style. Salesforce Knowledge only supports one column in article type layouts.
`style`	LayoutSectionStyle (enumeration of type string)	The style of the layout. Salesforce Knowledge only supports the value `OneColumn` which displays a one column page.

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).

LayoutItem

LayoutItem represents the valid values that define a layout item.

Field Name	Field Type	Description
field	string	The field name reference, for example MyField__c.

Declarative Metadata Sample Definition

The following is the definition of an ArticleType page layout:

```
<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
    <layoutSections>
        <customLabel>true</customLabel>
        <label>Description</label>
        <layoutColumns>
            <layoutItems>
                <field>description__c</field>
            </layoutItems>
            <layoutItems>
                <field>dateTime__c</field>
            </layoutItems>
        </layoutColumns>
        <style>OneColumn</style>
    </layoutSections>
    <layoutSections>
        <label>Data Sheet</label>
        <layoutColumns>
            <layoutItems>
                <field>file__c</field>
            </layoutItems>
        </layoutColumns>
        <style>OneColumn</style>
    </layoutSections>
</Layout>
```

SEE ALSO:

ArticleType

ArticleType CustomField

ArticleType CustomField

Represents the metadata associated with an article type custom field. Use this metadata type to create, update, or delete article type custom field definitions. This type extends the Metadata metadata type and inherits its fullName field.

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

```
MyArticleType__kav.MyCustomField__c
```

Declarative Metadata File Suffix and Directory Location

Custom fields are defined as part of the article type. ArticleType field names have a suffix of __c like other custom objects, and must be dot-qualified with the name of the article type to which they belong. See ArticleType for more information.

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` retrieves the files `objects/MyCustomObject__c.object`, `objects/Account.object__c.object`, and `objects/MyArticleType__kav.object`, each containing one custom field definition.

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

Version

ArticleTypes custom fields are available in API version 19.0 and later.

Fields for ArticleType

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

Note: If you create a knowledge validation rule, the errors always display at the top of the page, even if you add it beside the field. Therefore, write the errors descriptively so authors know how to satisfy the validation rule. For example, identify which field is causing the error. The Salesforce Classic user interface does not support field level error messages for articles.

Field Name	Field Type	Description
`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 "Object Relationships" in the Salesforce Help.
`description`	string	Description of the field.
`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 `BlankAsBlank` and `BlankAsZero`.

Field Name	Field Type	Description
`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 `create()` to see an example of this field specified for a call. This value cannot be `null`.
`inlineHelpText`	string	Represents the content of field-level help. For more information, see "Define Field-Level Help" in the Salesforce Help.
`label`	string	Label for the field. You cannot update the label for standard fields in Article Type such as Title, UrlName, Summary, etc.
`length`	int	Length of the field.
`picklist`	Picklist (Including Dependent Picklist)	If specified, the field is a picklist, and this field enumerates the picklist values and labels.
`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 MyObject that had a relationship to YourObject, the relationship name might be YourObjects.
`required`	boolean	Indicates whether the field requires a value on creation (`true`) or not (`false`).
`type`	FieldType	Required. Indicates the field type for the field. Valid values are: • `Checkbox` available in version 30.0 and later • `Currency` • `ArticleCurrency` • `Date` • `DateTime` • `Email` • `File` • `Formula` • `Html` • `Lookup` • `Number` • `Percent` • `Phone` • `Picklist` • `DependentPicklist` • `MultiselectPicklist` • `Text`

Field Name	Field Type	Description
		• `TextArea` • `LongTextArea` • `URL`
`visibleLines`	int	Indicates the number of lines displayed for the field.

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

SEE ALSO:

ArticleType

ArticleType Layout

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: By default, you can't deploy updates to an Apex class if there are one or more active jobs for that class. To deploy updates in this case, do one of the following.

- Cancel Apex jobs before deploying changes to Apex code. Reschedule the jobs after the deployment.
- Enable deployments with Apex jobs in the Salesforce user interface in the Deployment Settings page.

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.
`packageVersions`	PackageVersion[]	The list of installed managed package versions that are referenced by this Apex class. For more information about managed packages, see the Force.com Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.
`status`	ApexCodeUnitStatus (enumeration of type string)	The current status of the Apex class. The following string values are valid: • `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. Note: ApexCodeUnitStatus includes an `Inactive` option, but it is only supported for ApexTrigger; it is not supported for ApexClass.

PackageVersion

PackageVersion identifies a version of a managed package. A package version is a number that identifies the set of components uploaded in a package. The version number has the format *majorNumber.minorNumber.patchNumber* (for example, 2.1.3). The major and minor numbers increase to a chosen value during every major release. The *patchNumber* is generated and updated only for a patch release. It is available in API version 16.0 and later.

Field Name	Field Type	Description
namespace	string	Required. In a packaging context, a namespace prefix is a one to 15-character alphanumeric identifier that distinguishes your package and its contents from packages of other developers on AppExchange. Namespace prefixes are case-insensitive. For example, ABC and abc are not recognized as unique. Your namespace prefix must be globally unique across all Salesforce organizations. It keeps your managed package under your control exclusively. Salesforce automatically prepends your namespace prefix, followed by two underscores ("__"), to all unique component names in your Salesforce organization. A unique package component is one that requires a name that no other component has within Salesforce, such as custom objects, custom fields, custom links, s-controls, and validation rules. For more information about namespaces, see "Register a Namespace Prefix" in the Salesforce online help.
majorNumber	int	Required. The major number of the package version. A package version number has a *majorNumber.minorNumber* format.
minorNumber	int	Required. The minor number of the package version. A package version number has a *majorNumber.minorNumber* format.

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>36.0</apiVersion>
</ApexClass>
```

SEE ALSO:

ApexTrigger

ApexComponent

Represents a Visualforce component. For more information, see "Visualforce" in the Salesforce online help. This metadata type extends the MetadataWithContent component and shares its fields.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.component` for the page file. The accompanying metadata file is named *`ComponentName`*`-meta.xml`.

Visualforce components are stored in the `components` folder in the corresponding package directory.

Version

Visualforce components are available in API version 12.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`apiVersion`	double	The API version for this Visualforce component. Every component has an API version specified at creation. This field is available in API version 16.0 and later.
`content`	base64Binary	The component content. 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.
`description`	string	A description of what the component does.
`fullName`	string	The component developer name used as a 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`label`	string	Required. The label for this component.

Field Name	Field Type	Description
packageVersions	PackageVersion[]	The list of installed managed package versions that are referenced by this Visualforce component. Note: Package components and Visualforce custom component are distinct concepts. A package is comprised of many elements, such as custom objects, Apex classes and triggers, and custom pages and components. For more information about managed packages, see the Force.com Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.

SEE ALSO:

ApexPage

ApexPage

Represents a Visualforce page. For more information, see "Visualforce" in the Salesforce online help. This metadata type extends the MetadataWithContent component and shares its fields.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.page` for the page file. The accompanying metadata file is named *PageName*`-meta.xml`.

Visualforce pages are stored in the `pages` folder in the corresponding package directory.

Version

Visualforce pages are available in API version 11.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
apiVersion	double	Required. The API version for this page. Every page has an API version specified at creation. This field is available in API version 15.0 and later. If you set this field to a number lower than 15.0, it will be changed to 15.0.
content	base64Binary	The page content. 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

Field Name	Field Type	Description
		by a SOAP client. This field is inherited from the MetadataWithContent component.
`description`	string	A description of what the page does.
`fullName`	string	The page developer name used as a 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`availableInTouch`	boolean	Indicates if Visualforce tabs associated with the Visualforce page can be used in the Salesforce1 app. (Use of this field for Salesforce Touch is deprecated.). This field is available in API version 27.0 and later. Standard object tabs that are overridden with a Visualforce page aren't supported in Salesforce1, even if you set this field for the page. The default Salesforce1 page for the object is displayed instead of the Visualforce page.
`confirmationTokenRequired`	boolean	Indicates whether `GET` requests for the page require a CSRF confirmation token. This field is available in API version 28.0 and later. If you change this field's value from `false` to `true`, links to the page require a CSRF token to be added to them, or the page will be inaccessible.
`label`	string	Required. The label for this page.
`packageVersions`	PackageVersion[]	The list of installed managed package versions that are referenced by this Visualforce page. For more information about managed packages, see the Force.com Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.

Declarative Metadata Sample Definition

The following sample creates the `MyPage.page` page, and the corresponding `MyPage.page-meta.xml` metadata file.

`SampleApexPage.page` file:

```
<apex:page>
<h1>Congratulations</h1>
This is your new Page.
</apex:page>
```

`SampleApexPage.page-meta.xml`:

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

```
    <description>This is a sample Visualforce page.</description>
    <label>SampleApexPage</label>
</ApexPage>
```

SEE ALSO:

ApexComponent

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 "Manage Apex Triggers" in the Salesforce 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
`apiVersion`	double	Required. The API version for this trigger. Every trigger has an API version specified at creation.
`content`	base64	The Apex trigger definition. This field is inherited from the MetadataWithContent component.
`fullName`	string	The Apex trigger 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.

Field Name	Field Type	Description
packageVersions	PackageVersion[]	The list of installed managed package versions that are referenced by this Apex trigger. For more information about managed packages, see the Force.com Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.
status	ApexCodeUnitStatus (enumeration of type string)	Required. The current status of the Apex trigger. The following string values are valid: • Active - The trigger is active. • Inactive - The trigger is inactive, but not deleted. • Deleted - 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>36.0</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
appMenuItems	AppMenuItem[]	A list of menu items in the app menu.

AppMenuItem

Represents a menu item in the app menu.

Field Name	Field Type	Description
name	string	The API name of the item.
type	string	The type of application represented by this item. Acceptable values for AppSwitcher.appMenu are: • *ConnectedApp* • *CustomApplication* • *ServiceProvider* Acceptable values for Salesforce1.appMenu are: • *CustomApplication* • *CustomTab* • *StandardAppMenuItem*. The name for this item can be: – *MyDay* – *Feed* – *Tasks* – *Dashboards* – *Search* – *People* (available only when Chatter is enabled)

Field Name	Field Type	Description
		– *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>StandardAppMenuItem</type>
      </appMenuItem>
      <appMenuItem>
```

```
            <name>StandardItem2</name>
            <type>StandardAppMenuItem</type>
        </appMenuItem>
        <appMenuItem>
            <name>StandardItem3</name>
            <type>StandardAppMenuItem</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.

ApprovalProcess

Represents the metadata associated with an approval process. An approval process is an automated process your organization can use to approve records in Salesforce. An approval process specifies the steps necessary for a record to be approved and who must approve it at each step. A step can apply to all records included in the process, or just records that meet certain administrator-defined criteria. An approval process also specifies the actions to take when a record is approved, rejected, recalled, or first submitted for approval. This type extends the Metadata metadata type and inherits its `fullName` field.

Note:

- To use approval processes on Salesforce Knowledge articles with the Metadata API, the article type must be deployed. For article version (_kav) in approval processes, the supported action types are: Knowledge Action, Email Alert, Field Update, and Outbound Message.
- Send actions and approval processes for email drafts aren't supported in the Metadata API.
- The metadata doesn't include the order of active approval processes. You may need to reorder the approval processes in the destination organization after deployment.
- Before you implement an approval process for your organization, see "Considerations for Approval Processes" in the Salesforce Help.

File Suffix and Directory Location

ApprovalProcess components have the suffix `.approvalProcess` and are stored in the `approvalProcesses` folder.

Version

ApprovalProcess components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Required. Whether the approval process is active. After an approval process is activated, you can't add, delete, or change the order of the steps or change reject or skip behavior for that process, even if you make the process inactive.
`allowRecall`	boolean	Whether to allow submitters to recall approval requests. If set to `false`, only administrators can recall approval requests.
`allowedSubmitters`	ApprovalSubmitter[]	Required. An array of users who are allowed to submit records for approval.
`approvalPageFields`	ApprovalPageField	Specifies which fields to display on the approval page, where the approver goes to approve or reject the record. By default, the approval page displays the following: • `Name` field • `Owner` field (except for child objects) If you enable notifications in Salesforce1, keep in mind that approvers may view this list of fields on a mobile device. Select only the fields necessary for users to decide whether to approve or reject records.

Field Name	Field Type	Description
`approvalStep`	ApprovalStep[]	An array of approval step definitions.
`description`	string	Describes the approval process.
`emailTemplate`	string	Specifies which email template to use for approval requests. If not specified, the default email template is used. When an approval process assigns an approval request to a user, Salesforce automatically sends the user an approval request email. The email contains a link to the approval page in Salesforce, where the user can approve or reject the request and add comments.
`enableMobileDeviceAccess`	boolean	Whether users can access an external version of the approval page from any browser, including those on mobile devices, without logging in to Salesforce. Corresponds to `Security Settings` in the user interface. If set to `true`, approval steps can't have approvers of `type adhoc`. If set to `false`, approvers must log in to Salesforce to access the approval page.
`entryCriteria`	ApprovalEntryCriteria	Determines which records can enter the approval process. Exclude this field to allow all records to enter the approval process.
`finalApprovalActions`	ApprovalAction	Specifies which workflow actions to execute when all required approvals have been given for a record.
`finalApprovalRecordLock`	boolean	Whether to keep the record locked after it receives all necessary approvals. Default: `false`.
`finalRejectionActions`	ApprovalAction	Specifies which workflow actions to execute after a record enters the final rejection state.
`finalRejectionRecordLock`	boolean	Whether to keep the record locked after it's finally rejected. Default: `false`.
`initialSubmissionActions`	ApprovalAction	Specifies which workflow actions to execute when a record is initially submitted for approval.
`label`	string	Required. Name of the approval process.
`nextAutomatedApprover`	NextAutomatedApprover	Specifies a standard or custom user hierarchy field that can be used to automatically assign the approver for an approval step. If you exclude this field, then no approval step can use a user hierarchy field to automatically assign the approver.

Field Name	Field Type	Description
`postTemplate`	string	Post template to use for Approvals in Chatter. Chatter post approval notifications are only available for approval processes associated with an object that has been enabled for feed tracking.
`recallActions`	ApprovalAction	Specifies which workflow actions to execute when a pending approval request is withdrawn.
`recordEditability`	RecordEditabilityType (enumeration of type string)	Specifies which users can edit records that are pending approval. When a record is submitted for approval, it is automatically locked to prevent other users from editing it during the approval process. Valid values are: • `AdminOnly`—Records pending approval can be edited by: – Users with the "Modify All Data" permission – Users with the "Modify All" object-level permission for the given object • `AdminOrCurrentApprover`—Records pending approval can be edited by: – Users with the "Modify All Data" permission – Users with the "Modify All" object-level permission for the given object – The assigned approver, who must have edit access to the record through user permissions and the organization-wide sharing defaults for the given object
`showApprovalHistory`	boolean	Whether to add the Approval History related list to the approval page, which is where the approver can view the approval request details and approve or reject the record. The Approval History related list tracks a record through the approval process. If you also want to add the Approval History related list to record detail and edit pages, use the Salesforce user interface to customize the page layouts for the given object.

ApprovalSubmitter

Represents a user or set of users who can submit records for approval.

Field Name	Field Type	Description

`submitter`	string	Identifies a specific user or set of users who can submit records for approval. This field is required, except when the following types are specified and the `submitter` field is ignored:
		• `owner`
		• `creator`
		• `allInternalUsers`
		Example:
		`<allowedSubmitters>`
		`  <type>allInternalUsers</type>`
		`</allowedSubmitters>`
		`<allowedSubmitters>`
		`  <submitter>myGroup</submitter>`
		`  <type>group</type>`
		`</allowedSubmitters>`

`type`	ProcessSubmitterType (enumeration of type string)	Required. Type of user or set of users who can submit records for approval. Valid values are: • `group` • `role` • `user` • `roleSubordinates` • `roleSubordinatesInternal` • `owner` • `creator` • `partnerUser` • `customerPortalUser` • `portalRole` • `portalRoleSubordinates` • `allInternalUsers`—all Salesforce users in the organization

ApprovalPageField

Represents the selection of fields to display on the approval page, where an approver can view the approval request details and approve or reject the record.

Field Name	Field Type	Description
`field`	string[]	An array of fields that are displayed on the page for the approver to approve or reject the record.

ApprovalStep

Represents a step in the approval process. Approval steps assign approval requests to various users and define the chain of approval for a particular approval process. Each approval step specifies the attributes a record must have to advance to that approval step, the user who can approve requests for those records, and whether to allow the delegate of the approver to approve the requests. The first approval step in a process specifies the action to take if a record does not advance to that step. Subsequent steps in the process allow you to specify what happens if an approver rejects the request.

Note:

- The order of the `ApprovalStep` entries in the approval process definition determines the order in which the approval steps are executed.
- After an approval process is activated, you can't add, delete, or change the order of the steps or change reject or skip behavior for that process, even if you make the process inactive.
- There is a limit of 30 steps.

Field Name	Field Type	Description
`allowDelegate`	boolean	Whether to allow delegated approvers in this step of the approval process. A delegated approver is a user appointed by an assigned approver as an alternate for approval requests.
`approvalActions`	ApprovalAction	Specifies which workflow actions to execute when a record is approved in this step of the approval process.
`assignedApprover`	ApprovalStepApprover	Specifies the assigned approvers for this step of the approval process.
`description`	string	Describes the approval step.
`entryCriteria`	ApprovalEntryCriteria	Determines which records can enter this step of the approval process.
`ifCriteriaNotMet`	StepCriteriaNotMetType (enumeration of type string)	Specifies what to do for records that don't meet the entry criteria. Valid values are: • `ApproveRecord`—Approve the request and execute all final approval actions. • `RejectRecord`—Reject the request and execute all final rejection actions. This option is available only for the first step in the approval process. • `GotoNextStep`—Skip to the next approval step. If you select this option for the first approval step, and a record doesn't meet the entry criteria for any other step, the record is rejected.
`label`	string	Required. Name of the approval step.
`name`	string	Required. Unique name of the approval step. It must contain only underscores and alphanumeric characters, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. The requirement for uniqueness is only within the specific approval process.

Field Name	Field Type	Description
`rejectBehavior`	ApprovalStepRejectBehavior	Required, except for the first step in the approval process. Specifies what happens if the approver rejects the request during this approval step, unless it's the first step in the approval process. If the approver rejects the request in the first step in the approval process, the reject behavior is determined by the `finalRejectionActions`.
`rejectionActions`	ApprovalAction	Specifies which workflow actions to execute when a record is rejected in this step of the approval process.

ApprovalAction

Represents the actions that occur as a result of an approval process.

Field Name	Field Type	Description
`action`	WorkflowActionReference[]	An array of workflow actions to execute.

ApprovalStepApprover

Represents the assigned approvers for an approval step. There is a limit of 25 approvers per step.

Field Name	Field Type	Description
`approver`	Approver[]	An array of assigned approvers for this step of the approval process.
`whenMultipleApprovers`	RoutingType (enumeration of type string)	Specifies how to handle approval or rejection when multiple approvers are assigned to the step. Valid values are: • `Unanimous`—(Default) Require unanimous approval from all approvers for this step. The approval request for this step is rejected if any of the approvers reject the request. • `FirstResponse`—Approve or reject based on the first response.

Approver

Represents an assigned approver for an approval step.

Note: When specifying approvers, note the following:

- Make sure the assigned approver has access to read the records for the approval requests. For example, a user who does not have read access to an Expenses custom object will not be able to view expense approval requests.
- You can't use email approval response with approval processes in which the assigned approver is a queue.
- Approvers must have the "API Enabled" system permission to approve or reject approval requests via email.

- You can assign an approval request to the same user multiple times in a single step; however, Salesforce recognizes such redundancies and only requests a single approval from the user.
- After a record enters an approval step, the designated approvers for that step don't change even if the approval process returns to that step and the values of related user fields that designate the approvers have changed. For example, consider an approval process in which the first step requests approval from a user's manager. If the approval request is rejected in the second step and sent back to the first step, the user's original manager receives the approval request again even if the user's manager has changed.
- When the assigned approver is a queue:
  - Any queue member can approve or reject an approval request that is assigned to the queue.
  - Approval request emails are sent to the queue email address. If the queue is set up to send email to members, then approval request emails are also sent to the queue members, except those whose approval user preferences are set to never receive approval request emails.
  - When an approval request is assigned to a queue, each queue member's delegated approver also receives an approval request email notification.
  - Because email notifications to a queue aren't intended for an external audience, any instances of the merge field `{!ApprovalRequest.External_URL}` in the email template are sent as the equivalent internal URL.
  - Salesforce1 notifications for approval requests aren't sent to queues. For each approval step involving a queue, we recommend adding individual users as assigned approvers, so at least those individuals can receive the approval request notifications in Salesforce1. To have both queues and individual users as assigned approvers, select `Automatically assign to approver(s)` instead of `Automatically assign to queue` in the approval step.
  - When an approval request is rejected and returned to the previous approver and the previous approver was a queue, the approval request is assigned to the user who approved it instead of the queue.
  - The Approval History related list displays the queue name in the `Assigned To` column and the actual user who approved or rejected the approval request in the `Actual Approver` column.

Field Name	Field Type	Description
`name`	string	Identifies an assigned approver. This field is required, except when the `type` is one of the following and the `name` is ignored: • `adhoc` • `userHierarchyField`
`type`	NextOwnerType (enumeration of type string)	Combined with the specified `name`, this identifies an assigned approver. Valid values are: • `adhoc`—The approver for the step must be selected manually. For the first step, the submitter selects the approver. For the second and later steps, the approver for the previous step selects the approver. For this value, exclude the `name` field. • `user`—A user in your organization. For this value, enter a username for the `name` field. • `userHierarchyField`—A user specified in a standard or custom user hierarchy field, such as the standard `Manager` field. For this value, exclude the `name` field. The user hierarchy field must be defined in the `nextAutomatedApprover` for the approval process. • `relatedUserField`—A user specified in a user lookup field on the submitted record, such as the `Last Modified By` field. For this value, enter the name of the user lookup field for the `name` field.

Field Name	Field Type	Description
		• `queue`—Automatically assign to a queue. For this value, enter the name of the queue for the `name` field.

ApprovalEntryCriteria

Represents the criteria that records must meet to enter the approval process or an approval step. Specify either filter criteria or a formula, but not both.

Field Name	Field Type	Description
`booleanFilter`	string	Filter logic for `criteriaItems`. Exclude this field if you enter a `formula`.
`criteriaItems`	FilterItem[]	Filter criteria that a record must meet to enter the approval process or approval step. Approval processes don't support `valueField` entries in filter criteria.
`formula`	string	Formula that must evaluate to true for a record to enter the approval process or approval step.

ApprovalStepRejectBehavior

Represents what happens if the approver rejects the request during this approval step, unless it's the first step in the approval process. For the first step in the approval process, the reject behavior is determined by the approval process's final rejection actions.

Field Name	Field Type	Description
`type`	StepRejectBehaviorType (enumeration of type string)	Valid values are: • `RejectRequest`—Automatically rejects the request completely regardless of any previous steps that were approved. Salesforce performs all rejection actions specified for this step and all final rejection actions. • `BackToPrevious`—Automatically rejects the request and returns the approval request to the previous approver. Salesforce performs all rejection actions specified for this step. Not allowed in the first step of the approval process.

NextAutomatedApprover

Represents the user hierarchy field to use as the next automated approver for the approval process. If defined, the user specified in the hierarchy field can be automatically assigned as the approver in one or more approval steps.

Field Name	Field Type	Description
`useApproverFieldOfRecordOwner`	boolean	Required. Whether the first executed approval step should use the specified `userHierarchyField` in the record owner's user record—instead of the submitter's user record—as the approver. All

Field Name	Field Type	Description
		remaining steps use the specified `userHierarchyField` in the user record of the preceding step's approver.
`userHierarchyField`	string	Required. Standard or custom user hierarchy field whose value specifies which user to assign as the approver. For example, the standard `Manager` hierarchy field can be used to assign approvers for employee PTO (paid time off) requests. Make sure the assigned approver has access to read the records for the approval requests. For example, a user who does not have read access to an Expenses custom object will not be able to view expense approval requests.

Declarative Metadata Sample Definition

The following is an example of an ApprovalProcess component:

```
<?xml version="1.0" encoding="UTF-8"?>
<ApprovalProcess xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>false</active>
    <allowRecall>false</allowRecall>
    <allowedSubmitters>
        <type>owner</type>
    </allowedSubmitters>
    <allowedSubmitters>
        <submitter>USSalesRep</submitter>
        <type>role</type>
    </allowedSubmitters>
    <allowedSubmitters>
        <submitter>MarketingGroup</submitter>
        <type>group</type>
    </allowedSubmitters>
    <allowedSubmitters>
        <submitter>kcooper@example.com</submitter>
        <type>user</type>
    </allowedSubmitters>
    <approvalPageFields>
        <field>Name</field>
        <field>Owner</field>
        <field>MyLeadCustomField__c</field>
        <field>Address</field>
    </approvalPageFields>
    <approvalStep>
        <allowDelegate>false</allowDelegate>
        <approvalActions>
            <action>
                <name>LeadApprovedTask1</name>
                <type>Task</type>
            </action>
            <action>
```

```
            <name>LeadApprovedTask2</name>
            <type>Task</type>
        </action>
    </approvalActions>
    <assignedApprover>
        <approver>
            <type>adhoc</type>
        </approver>
    </assignedApprover>
    <label>Step1</label>
    <name>Step1</name>
    <rejectionActions>
        <action>
            <name>LeadRejectedTask</name>
            <type>Task</type>
        </action>
    </rejectionActions>
</approvalStep>
<approvalStep>
    <allowDelegate>false</allowDelegate>
    <assignedApprover>
        <approver>
            <type>userHierarchyField</type>
        </approver>
    </assignedApprover>
    <entryCriteria>
        <criteriaItems>
            <field>Lead.CreatedDate</field>
            <operation>greaterThan</operation>
            <value>3/25/2013</value>
        </criteriaItems>
        <criteriaItems>
            <field>User.IsActive</field>
            <operation>notEqual</operation>
            <value>true</value>
        </criteriaItems>
    </entryCriteria>
    <ifCriteriaNotMet>ApproveRecord</ifCriteriaNotMet>
    <label>Step2</label>
    <name>Step2</name>
    <rejectBehavior>
        <type>RejectRequest</type>
    </rejectBehavior>
</approvalStep>
<approvalStep>
    <allowDelegate>true</allowDelegate>
    <assignedApprover>
        <approver>
            <name>MarketingTeamQueue</name>
            <type>queue</type>
        </approver>
        <approver>
            <name>LastModifiedBy</name>
            <type>relatedUserField</type>
```

```
            </approver>
            <approver>
                <name>awheeler@example.com</name>
                <type>user</type>
            </approver>
            <whenMultipleApprovers>FirstResponse</whenMultipleApprovers>
        </assignedApprover>
        <entryCriteria>
            <formula>CONTAINS( MyLeadCustomField__c , 'Salesforce')</formula>
        </entryCriteria>
        <label>Step3</label>
        <name>Step3</name>
        <rejectBehavior>
            <type>BackToPrevious</type>
        </rejectBehavior>
    </approvalStep>
    <emailTemplate>MyFolder/LeadsNewassignmentnotification</emailTemplate>
    <enableMobileDeviceAccess>false</enableMobileDeviceAccess>
    <entryCriteria>
        <criteriaItems>
            <field>Lead.AnnualRevenue</field>
            <operation>greaterThan</operation>
            <value>10500</value>
        </criteriaItems>
        <criteriaItems>
            <field>Lead.MyLeadCustomField__c</field>
            <operation>equals</operation>
            <value>Salesforce</value>
        </criteriaItems>
    </entryCriteria>
    <finalApprovalActions>
        <action>
            <name>LeadEmailContacted</name>
            <type>Alert</type>
        </action>
    </finalApprovalActions>
    <finalApprovalRecordLock>true</finalApprovalRecordLock>
    <finalRejectionActions>
        <action>
            <name>ProcessRejectedMessageAction</name>
            <type>OutboundMessage</type>
        </action>
    </finalRejectionActions>
    <finalRejectionRecordLock>false</finalRejectionRecordLock>
    <initialSubmissionActions>
        <action>
            <name>LeadFieldUpdate</name>
            <type>FieldUpdate</type>
        </action>
        <action>
            <name>NewLeadEmail</name>
            <type>Alert</type>
        </action>
    </initialSubmissionActions>
```

```
    <label>SampleProcess</label>
    <nextAutomatedApprover>
        <useApproverFieldOfRecordOwner>false</useApproverFieldOfRecordOwner>
        <userHierarchyField>customlookupuserfield__c</userHierarchyField>
    </nextAutomatedApprover>
    <postTemplate>MyPostTemplate</postTemplate>
    <recallActions>
        <action>
            <name>ProcessRecalledMessageAction</name>
            <type>OutboundMessage</type>
        </action>
    </recallActions>
    <recordEditability>AdminOnly</recordEditability>
    <showApprovalHistory>false</showApprovalHistory>
</ApprovalProcess>
```

AssignmentRules

Represents assignment rules that allow you to automatically route cases to the appropriate users or queues. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object. The `package.xml` syntax for accessing all assignment rules for all objects is:

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

All rules for a specific object uses a similar syntax without the wildcard. For example, all assignment rules for the Case object would use this syntax:

```
    <types>
        <members>Case</members>
        <name>AssignmentRules</name>
    </types>
```

You can also access specific assignment rules for an object. The following example only accesses the “samplerule” and “newrule” assignment rules on the Case object. Notice that for this example the type name syntax is `AssignmentRule` and not `AssignmentRules`.

```
    <types>
        <members>Case.samplerule</members>
        <members>Case.newrule</members>
        <name>AssignmentRule</name>
    </types>
```

File Suffix and Directory Location

Assignment rules for an object have the suffix `.assignmentRules` and are stored in the `assignmentRules` folder. For example, all Case assignment rules are stored in the `Case.assignmentRules` file.

Version

AssignmentRules components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`assignmentRule`	AssignmentRule[]	Represents the definitions of the named assignment rules.

AssignmentRule

Specifies whether the rule is active or not and its definition. Rules are processed in the order they appear within the AssignmentRules container.

Field Name	Field Type	Description
`active`	boolean	Indicates whether the assignment rule is active (`true`) or not (`false`).
`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 `create()` to see an example of this field specified for a call. This value cannot be `null`.
`ruleEntry`	`RuleEntry[]`	Represents the type and description for the assignment rule.

RuleEntry

Represents the fields used by the rule.

Field Name	Field Type	Description
`assignedTo`	string	The name of the user or queue the item is assigned to.
`assignedToType`	`AssignToLookupValueType` (enumeration of type string)	Valid values are: • `User` • `Queue`
`booleanFilter`	string	Advanced filter conditions that were specified for the rule.
`criteriaItems`	`FilterItem[]`	The items in the list that define the assignment criteria.
`formula`	string	The validation formula. Note: Specify either `formula` or `criteriaItems`, but not both fields.

Field Name	Field Type	Description
`notifyCcRecipients`	boolean	Specifies whether email addresses included on the Cc line of an incoming Email-to-Case or Web-to-Lead message should be included on the Cc line of the auto-response to that message (`true`) or not (`false`). Available in API version 32.0 and later.
`overrideExistingTeams`	boolean	Specifies whether the case team should be reset when the assignment is done `true`) or if the current team is added to the case instead of replacing the previous team (`false`).
`team`	string[]	The name of the case team. It may occur 0 or more times.
`template`	string	Specifies the template to use for the email that is automatically sent to the designated recipient.

Declarative Metadata Sample Definition

The following is an example file showing two assignment rules on the Case object:

```
<AssignmentRules xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <assignmentRule>
        <fullName>samplerule</fullName>
        <active>false</active>
        <ruleEntry>
            <assignedTo>testUser@org.com</assignedTo>
            <assignedToType>User</assignedToType>
            <criteriaItems>
                <field>Case.IsEscalated</field>
                <operation>equals</operation>
                <value>True</value>
            </criteriaItems>
            <template>emailtemplate</template>
        </ruleEntry>
    </assignmentRule>
    <assignmentRule>
        <fullName>Another samplerule</fullName>
        <active>false</active>
        <ruleEntry>
            <assignedTo>otherUser@org.com</assignedTo>
            <assignedToType>User</assignedToType>
            <criteriaItems>
                <field>Case.IsEscalated</field>
                <operation>equals</operation>
                <value>False</value>
            </criteriaItems>
            <template>emailtemplate</template>
        </ruleEntry>
    </assignmentRule>
</AssignmentRules>
```

AuraDefinitionBundle

Represents a Lightning definition bundle. A bundle contains a Lightning definition and all its related resources. The definition can be a component, application, event, interface, or a tokens collection.

File Suffix and Directory Location

A Lightning bundle is a folder that contains definition files. Unlike other metadata components, an AuraDefinitionBundle component isn't represented by a single component file but instead by a collection of component definition files. Each definition file represents a resource in a bundle, such as markup, applications, code files (including controllers and helpers), events, documentation, and interfaces. For example, this directory structure shows the hierarchy of the folders and files for two bundles: `bundle1` and `bundle2`.

```
aura
    bundle1
        bundle1.cmp
        bundle1Controller.js
    bundle2
        bundle2.app
        bundle2.cmp
        bundle2Controller.js
        bundle2.auradoc
```

Lightning bundles must be under a top-level folder that's named `aura`. Each bundle must have its own subfolder under the `aura` folder. The name of each definition file must start with the bundle name.

A bundle doesn't have a suffix. Definition files can have one of these suffixes:

Suffix	Component Type
`.app`	Application
`.cmp`	Component
`.design`	Design
`.evt`	Event
`.intf`	Interface
`.js`	Controller, Helper, or Renderer
`.svg`	SVG image
`.css`	Style
`.auradoc`	Documentation
`.tokens`	Tokens collection

Each bundle can have only one file each with a suffix of `.app`, `.cmp`, `.design`, `.evt`, `.intf`, or `.tokens`.

Version

AuraDefinitionBundle components are available in API version 32.0 and later.

Design and SVG components are available in API version 33.0 and later.

Special Access Rules

Definitions can be created only in organizations with defined namespaces.

Fields

Field Name	Field Type	Description
`apiVersion`	double	The API version for this definition bundle. When you create an Aura bundle, you can specify the API version to save it with. Available in API version 35.0 and later.
`controllerContent`	base64Binary	The content of a JavaScript client-side controller.
`description`	string	The specification of the Aura bundle. Available in API version 35.0 and later.
`designContent`	base64Binary	The content of a design definition. Only valid inside a component bundle.
`documentationContent`	base64Binary	The content of a documentation definition.
`helperContent`	base64Binary	The content of a JavaScript helper.
`markup`	base64Binary	The content of the markup for a definition.
`modelContent`	base64Binary	Deprecated. Do not use.
`packageVersions`	PackageVersion[]	The list of installed managed package versions that this Aura definition bundle references. Available in API version 35.0 and later.
`rendererContent`	base64Binary	The content of a JavaScript client-side renderer.
`styleContent`	base64Binary	The CSS for the definition.
`SVGContent`	base64Binary	The SVG image for the definition.
`testsuiteContent`	base64Binary	Reserved for future use.
`type`	AuraBundleType (enumeration of type string)	The definition type. Valid values are: • `Application` • `Component` • `Event` • `Interface` • `Tokens`

Declarative Metadata Sample Definition

This example shows the directory structure of an AuraDefinitionBundle component.

```
aura
    sampleCmp
        sampleCmp.cmp
        sampleCmpController.js
```

The following samples show the contents of the metadata definition files that correspond to the sample `aura` directory.

Content of `sampleCmp.cmp`:

```
<aura:component>
    <aura:attribute name="val1" type="String" default="Value"/>
    <aura:attribute name="val2" type="String" />
    <aura:handler name="init" value="{!this}" action="{!c.myAction}"/>
 <ui:outputText value='Hello world!'/>
 <ui:outputText value='{!v.val1}'/>
 <ui:outputText value='{!v.val2}'/>
</aura:component>
```

Content of `sampleCmpController.js`:

```
({
 myAction : function(component) {
  component.set('v.val1','Value1');
  component.set('v.val2','Value2');
    }
})
```

This `package.xml` references the definitions of all Lightning components that are present in the `sampleCmp` bundle.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>sampleCmp</members>
        <name>AuraDefinitionBundle</name>
    </types>
    <version>36.0</version>
</Package>
```

AuthProvider

Represents an authentication provider (or auth provider) in your organization. An auth provider enables users to log in to your Salesforce organization using their login credentials from an external service provider such as Facebook© or Janrain©. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Authentication providers are stored in the `authproviders` directory. The file name matches the URL suffix and the extension is `.authprovider`. For example, an auth provider with URL suffix `FacebookProvider` is stored in `authproviders/FacebookProvider.authprovider`.

Version

Authentication providers are available in API version 27.0 and later.

Special Access Rules

Only users with the "Customize Application" and "Manage AuthProviders" permissions can access this object.

Fields

Field Name	Field Type	Description
`authorizeUrl`	string	Required, but only if `providerType` is `OpenIdConnect`. The OAuth authorization endpoint URL. Used only with OpenID Connect authentication providers. Available in API version 29.0 and later. In API version 33.0 and later, the behavior of this field changed to support the Salesforce-managed auth provider configuration, which allows Salesforce to manage the value for Facebook, Salesforce, LinkedIn, Twitter or Google authentication. For more information, see the Usage section.
`consumerKey`	string	Required. The app's key that is registered at the third-party single sign-on provider. In API version 33.0 and later, the behavior of this field changed to support the Salesforce-managed auth provider configuration, which allows Salesforce to manage the value for Facebook, Salesforce, LinkedIn, Twitter or Google authentication. For more information, see the Usage section.
`consumerSecret`	string	Required. The consumer secret of the app that is registered at the third-party single sign-on provider. This field cannot be updated. When using `create()` this field must be encrypted. To create an encrypted form of the consumer secret from plain text: 1. Create an auth provider with the `consumerSecret` plain text value. 2. Save the auth provider. 3. Create an outbound change set that includes the auth provider component. The new change set .xml file will have an entry in the form `<consumerSecret>++XYZ++</consumerSecret>` where ++XYZ++ is the encrypted secret. In API version 33.0 and later, the behavior of this field changed to support the Salesforce-managed auth provider configuration, which allows Salesforce to manage the value for Facebook, Salesforce, LinkedIn, Twitter or Google authentication. For more information, see the Usage section.

Field Name	Field Type	Description
customMetadataTypeRecord	string	Required, but only with custom authentication provider plug-ins, when ProviderType is Custom. The API name of the authentication provider. Available in API version 36.0 and later.
defaultScopes	string	Required, but only if providerType is OpenIdConnect The scopes to be sent with the authorization request, if not specified when a flow is started. Used only with OpenID Connect authentication providers. Available in API version 29.0 and later. In API version 33.0 and later, the behavior of this field changed to support the Salesforce-managed auth provider configuration, which allows Salesforce to manage the value for Facebook, Salesforce, LinkedIn, Twitter or Google authentication. For more information, see the Usage section.
DeveloperName	string	Required. Used when referring to the auth provider from a program.
errorUrl	string	A custom error URL for the provider to use to report any errors.
executionUserId	string	The user that runs the Apex handler class. The user must have the "Manage Users" permission. A user is required if you specify a registration handler class.
friendlyName	string	Required. A user-friendly name for the provider.
iconUrl	string	The path to an icon to use as a button on the login page for a community. Users click the button to log in to a community with the associated authentication provider, such as Twitter© or LinkedIn©. Available in API version 32.0 and later.
idTokenIssuer	string	Only available if providerType is OpenIdConnect. This value identifies the source of the authentication token in the form *https: URI*. Used only with OpenID Connect authentication providers. If provided, Salesforce validates the returned id_token value. The OpenID Connect specification requires an id_token value to be returned with the access_token value. Available in API version 30.0 and later.
logoutUrl	string	Provides a specific destination for users after they log out if they authenticated using the single sign-on flow. The URL must be fully qualified with an http or https prefix, such as https://acme.my.salesforce.com. Available in API version 33.0 and later.
plugin	string	An existing Apex class that implements the Auth.AuthProviderPlugin interface. Available in API version 36.0 and later.
providerType	AuthProviderType (enumeration of type string)	Required. The third-party single sign-on provider to use. Valid values are: • Facebook • Google • Salesforce • Janrain

Field Name	Field Type	Description
		• `LinkedIn` (Available in API version 32.0 and later.) • `Twitter` (Available in API version 32.0 and later.) • `OpenIdConnect` (Available in API version 29.0 and later.) Note: This type requires values for the following fields: – `authorizeUrl` – `defaultScopes` – `tokenUrl` – `userInfoUrl` • `MicrosoftACS` Microsoft Access Control Service typically provides authentication for a Microsoft Office 365 service like SharePoint® Online. (Available in API version 31.0 and later.) • `GitHub`—Use the `GitHub` provider to log in users of your Force.com app to GitHub using OAuth. When logged in to GitHub, your app can make calls to GitHub APIs. The `GitHub` provider isn't available as a single sign-on provider, which means users can't log in to your Salesforce organization using their GitHub login credentials. (Available in API version 35.0 and later.) • `Custom`—A provider configured with a custom authentication provider plug-in. (Available in API version 36.0 and later.)
`registrationHandler`	string	An existing Apex class that implements the `Auth.RegistrationHandler` interface.
`sendAccessTokenInHeader`	boolean	Required only if `providerType` is `OpenIdConnect`. When `true`, the access token is sent to the `userInfoUrl` in a header instead of a query string. Used only with OpenID Connect authentication providers. Available in API version 30.0 and later.
`sendClientCredentialsInHeader`	boolean	Required only if `providerType` is `OpenIdConnect`. When `true`, the client credentials are sent in a header, instead of a query string, to the `tokenUrl`. The credentials are in the standard OpenID Connect Basic Credentials header form, which is `Basic <token>`, where `<token>` is the base64-encoded string `"clientkey:clientsecret"`. Used only with OpenID Connect authentication providers. Available in API version 30.0 and later.
`tokenUrl`	string	Required, but only if `providerType` is `OpenIdConnect`. The OAuth token endpoint URL. Used only with OpenID Connect authentication providers. Available in API version 29.0 and later. In API version 33.0 and later, the behavior of this field changed to support the Salesforce-managed auth provider configuration, which allows Salesforce to manage the value for Facebook, Salesforce, LinkedIn, Twitter or Google authentication. For more information, see the Usage section.

Field Name	Field Type	Description
`userInfoUrl`	string	Required, but only if `providerType` is `OpenIdConnect`. The OpenID Connect endpoint URL. Used only with OpenID Connect authentication providers. Available in API version 29.0 and later. In API version 33.0 and later, the behavior of this field changed to support the Salesforce-managed auth provider configuration, which allows Salesforce to manage the value for Facebook, Salesforce, LinkedIn, Twitter or Google authentication. For more information, see the Usage section.

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthProvider xmlns="http://soap.sforce.com/2006/04/metadata">
    <consumerKey>yourappkey</consumerKey>
    <consumerSecret>PwdVxXjzu3NCZ3MD4He+wA==</consumerSecret>
    <executionUser>admin@your.org</executionUser>
    <friendlyName>FacebookAuthProvider</friendlyName>
    <providerType>Facebook</providerType>
    <registrationHandler>RegistrationHandler</registrationHandler>
</AuthProvider>
```

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

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

Usage

For API version 33.0 and later. when implementing the Salesforce-managed authentication provider configuration, you can have Salesforce create and manage the following fields for you:

- `authorizeUrl`
- `consumerKey`
- `consumerSecret`
- `defaultScopes`
- `tokenURL`
- `userInfoUrl`

To configure a Salesforce-managed authentication provider, set up a Facebook, Salesforce, LinkedIn, Twitter or Google auth provider and leave all of the listed fields blank. Salesforce automatically creates values for you. However, if you provide a value for any of these fields, then `consumerKey` and `consumerSecret` must also be specified.

AutoResponseRules

Represents an auto-response rule that sets conditions for sending automatic email responses to lead or case submissions based on the attributes of the submitted record. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object. The `package.xml` syntax for accessing all auto-response rules for all objects is:

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

All rules for a specific object uses a similar syntax without the wildcard. For example, all auto-response rules for the Case object would use this syntax:

```
<types>
    <members>Case</members>
    <name>AutoResponseRules</name>
</types>
```

You can also access specific auto-response rules for an object. The following example only accesses the “samplerule” and “newrule” auto-response rules on the Case object. Notice that for this example the type name syntax is `AutoResponseRule` and not `AutoResponseRules`.

```
<types>
    <members>Case.samplerule</members>
    <members>Case.newrule</members>
    <name>AutoResponseRule</name>
</types>
```

File Suffix and Directory Location

AutoResponseRules for an object have the suffix `.autoResponseRules` and are stored in the `autoResponseRules` folder. For example, all Case auto-response rules are stored in the `Case.autoResponseRules` file.

Version

AutoResponseRules components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`autoresponseRule`	AutoResponseRule[]	Represents the definitions of the named auto-response rules.

AutoResponseRule

Represents whether a rule is active or not and the order in which the entry is processed in the rule.

Field Name	Field Type	Description
`active`	boolean	Indicates whether the autoresponse rule is active (`true`) or not (`false`).
`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 `create()` to see an example of this field specified for a call. This value cannot be `null`.
`ruleEntry`	RuleEntry[]	Represents the type and description for the auto-response rule.

RuleEntry

Represents the fields used by the rule.

Field Name	Field Type	Description
`booleanFilter`	string	Advanced filter conditions that were specified for the rule.
`criteriaItems`	`FilterItem[]`	The items in the list that define the assignment criteria.
`formula`	string	The validation formula. Note: Specify either `formula` or `criteriaItems`, but not both fields.
`replyToEmail`	string	The email address that appears in the reply-to header.
`senderEmail`	string	The email address of the person or queue sending the email notification.
`senderName`	string	The name of the person or queue sending the email notification.
`template`	string	Specifies the template to use for the email that is automatically sent to the designated recipient.

Declarative Metadata Sample Definition

The following is an example AutoResponseRules component:

```
<AutoResponseRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <autoResponseRule>
        <fullName>ajbdeploytest2</fullName>
        <active>false</active>
        <ruleEntry>
            <criteriaItems>
                <field>Case.Description</field>
                <operation>contains</operation>
```

```
                <value>testing</value>
            </criteriaItems>
            <senderEmail>test@test.org</senderEmail>
            <senderName>tester name j</senderName>
            <replyToEmail>test@@test.org</replyToEmail>
            <template>emailtemplate</template>
        </ruleEntry>
    </autoResponseRule>
</AutoResponseRules>
```

CallCenter

Represents the Call Center definition used to integrate Salesforce 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 `displayName` field in Call Center setup page.
`internalNameLabel`	string	The label of the `internalName` 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 165	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>
        <items>
            <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>
```

Certificate

Represents a certificate used for digital signatures which verify that requests are coming from your org. Certificates are used for either authenticated single sign-on with an external website, or when using your org as an identity provider. This metadata type extends the MetadataWithContent component and shares its fields.

File Suffix and Directory Location

Certificate components have the suffix `.certificate` and are stored in the `certificate` folder.

Version

Certificate components are available in API version 36.0 and later.

Fields

Field Name	Field Type	Description
`caSigned`	boolean	Required. Indicates whether this certificate is signed by the issuer (true) or not (false).
`keySize`	int	Required. Certificate keys can be either 2048 bits or 4096 bits. A certificate with 4096-bit keys lasts two years, and a certificate with 2048-bit keys lasts one year. Certificates with 2048-bit keys are faster than certificates with 4096-bit keys.
`masterLabel`	string	Required. A user-friendly name for the certificate that appears in the Salesforce user interface, such as in Certificate and Key Management. Limit: 64 characters.

Declarative Metadata Sample Definition

The following is an example of a Certificate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Certificate xmlns="http://soap.sforce.com/2006/04/metadata">
<caSigned>true</caSigned>
<keySize>4096</keySize>
<masterLabel>My Certificate Name</masterLabel>
</Certificate>
```

Community (Zone)

Note: Starting with the Summer '13 release, Chatter Answers and Ideas "communities" have been renamed to "zones." In API version 28, the API object label has changed to `Zone`, but the API type is still `Community`.

Represents a zone that contains Ideas or Chatter Answers objects. Zones are shared by the Ideas, Answers, and Chatter Answers features, allowing you to view and create zones from those locations. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: When `enableChatterAnswers` is set to `false`, values specified for the following fields are ignored and not saved: `communityFeedPage`, `emailFooterDocument`, `emailHeaderDocument`, `enablePrivateQuestions`, `emailNotificationUrl`, and `site`.

File Suffix and Directory Location

Zones have the suffix `community` and are stored in the `communities` folder.

Version

Community (Zone) components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Indicates whether the zone is active (`true`) or not (`false`).
`communityFeedPage`	string	The Visualforce page that hosts the zone's feeds. This field is available when Chatter Answers is enabled in the organization.
`description`	string	The description of the zone.
`emailFooterDocument`	string	The text or HTML file that incorporates your organization's branding into the footer of email notifications. This field is available when Chatter Answers is enabled in the organization.
`emailHeaderDocument`	string	The text or HTML file that incorporates your organization's branding into the header of email notifications. This field is available when Chatter Answers is enabled in the organization.

Field Name	Field Type	Description
`emailNotificationUrl`	string	The URL that's included in email notifications. This field is available when Chatter Answers is enabled in the organization. This field replaces `portalEmailNotificationUrl` in API version 28.0 and later.
`enableChatterAnswers`	boolean	Indicates whether the zone has Chatter Answers enabled (`true`) or not (`false`). This field is available when Chatter Answers is enabled in the organization.
`enablePrivateQuestions`	boolean	Indicates whether Chatter Answers questions can be escalated to cases (`true`) or not (`false`). This field is available when Chatter Answers is enabled in the organization.
`expertsGroup`	string	The name of the public group that act as experts in the zone. This field is available when either Ideas or Answers are enabled in the organization.
`portal`	string	The name of the portal in which to display the zone.
`portalEmailNotificationUrl`	string	The portal URL that's included in email notifications. This field is available when Chatter Answers is enabled in the organization. This field has been replaced by `emailNotificationUrl` in API version 28.0 and later.
`reputationLevels`	ReputationLevels	The fields that define the points and name of each reputation level you define. You can create up to 25 reputation levels per zone.
`showInPortal`	boolean	Indicates whether the zone is available to all portals (`true`) or not available to any portals (`false`).
`site`	string	The name of the site for the zone. This field is available when Chatter Answers is enabled in the organization.

ReputationLevels

Represents the points and reputation label that displays on hover over a user's photo in the feed.

Field Name	Field Type	Description
`chatterAnswersReputationLevels`	ChatterAnswersReputationLevel []	Contains the name and value pair that describes the reputation level for Chatter Answers. Available in API version 28.0 and later.
`ideaReputationLevels`	IdeaReputationLevel	Contains the name and value pair that describes the reputation for Ideas. Available in API version 28.0 and later.

ChatterAnswersReputationLevel

Represents the reputation name and the number of points for that level for Chatter Answers.

Field Name	Field Type	Description
`name`	string	The name of the reputation level, for example, "Expert."

Field Name	Field Type	Description
value	int	The minimum number of points for the reputation level.

IdeaReputationLevel

Represents the reputation name and the number of points for that level for Ideas. Available in API version 28.0 and later.

Field Name	Field Type	Description
name	string	The name of the reputation level, for example, “Expert.”
value	int	The minimum number of points for the reputation level.

Declarative Metadata Sample Definition

The following is the definition of a community (zone) component:

```
<?xml version="1.0" encoding="UTF-8"?>
<Community xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <communityFeedPage>communityWithHeaderAndFooter_main</communityFeedPage>
    <description>Metadata Test</description>
    <emailFooterDocument>sampleFolder/emailFooter.html</emailFooterDocument>
    <emailHeaderDocument>sampleFolder/emailHeader.html</emailHeaderDocument>
    <enableChatterAnswers>true</enableChatterAnswers>
    <enablePrivateQuestions>true</enablePrivateQuestions>
    <expertsGroup>CommunityExperts</expertsGroup>
    <portal>Customer Portal</portal>
    <emailNotificationUrl>http://yourURL</emailNotificationUrl>
<reputationLevels>
        <chatterAnswersReputationLevels>
            <name>Newbie</name>
            <value>0</value>
        </chatterAnswersReputationLevels>
        <chatterAnswersReputationLevels>
            <name>Smartie</name>
            <value>500</value>
        </chatterAnswersReputationLevels>
        <chatterAnswersReputationLevels>
            <name>Pro</name>
            <value>2000</value>
        </chatterAnswersReputationLevels>
        <chatterAnswersReputationLevels>
            <name>All Star</name>
            <value>5000</value>
        </chatterAnswersReputationLevels>
        <ideaReputationLevels>
            <name>Observer</name>
            <value>0</value>
        </ideaReputationLevels>
        <ideaReputationLevels>
```

```
            <name>Contributor</name>
            <value>100</value>
        </ideaReputationLevels>
        <ideaReputationLevels>
            <name>Influencer</name>
            <value>400</value>
        </ideaReputationLevels>
        <ideaReputationLevels>
            <name>Thought Leader</name>
            <value>1500</value>
        </ideaReputationLevels>
    </reputationLevels>
    <showInPortal>true</showInPortal>
    <site>ChatterAnswersSite</site>
</Community>
```

ConnectedApp

Represents a connected app configuration. A connected app integrates an application with Salesforce using APIs. Connected apps use standard SAML and OAuth protocols to authenticate, provide Single Sign-On, and provide tokens for use with Salesforce APIs. In addition to standard OAuth capabilities, connected apps allow administrators to set various security policies and have explicit control over who may use the corresponding applications. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ConnectedApp components have the suffix `.connectedapp` and are stored in the `connectedapps` folder.

Version

ConnectedApp components are available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`attributes`	ConnectedAppAttribute	A custom attribute of the connected app.
`canvasConfig`	ConnectedAppCanvasConfig	The configuration options of the connected app if it's exposed as a canvas app.
`contactEmail`	string	The email address Salesforce should use for contacting you or your support team.
`contactPhone`	string	The phone number for Salesforce to use in case we need to contact you.
`description`	string	An optional description for your application.
`iconUrl`	string	Reserved for future use.

Field Name	Field Type	Description
`infoUrl`	string	An optional URL for a Web page with more information about your application.
`ipRanges`	ConnectedAppIpRange	Specifies the ranges of IP addresses that can access the app without requiring the user to authenticate with the connected app.
`label`	string	The name of the app.
`logoUrl`	string	An optional application logo. The logo appears with the application's entry in the list of apps and on the consent page the user sees when authenticating. The URL must use HTTPS, and the logo can't be larger than 125 pixels high or 200 pixels wide. The default logo is a cloud.
`mobileStartUrl`	string	Users are directed to this URL after they've authenticated when the app is accessed from a mobile device. If you don't give a URL, the user is sent to the application's default start page after authentication completes. If the connected app that you're creating is a canvas app, then you don't need to enter a value for this field. The Canvas App URL field contains the URL that gets called for the connected app.
`oauthConfig`	ConnectedAppOauthConfig	Specifies how your application communicates with Salesforce.
`plugin`	string	The name of a custom Apex class that extends `Auth.ConnectedAppPlugin` to customize the behavior of the app.
`samlConfig`	ConnectedAppSamlConfig	Controls how the app uses single sign-on.
`startUrl`	string	If the app is not accessed from a mobile device, users are directed to this URL after they've authenticated. If you don't give a URL, the user is sent to the application's default start page after authentication completes. If the app is accessed from a mobile device, see `mobileStartUrl`. If the connected app that you're creating is a canvas app, then you don't need to enter a value for this field. The Canvas App URL field contains the URL that gets called for the connected app.

ConnectedAppAttribute

Represents the field names that make up a custom attribute when using SAML with a ConnectedApp. These values should be tailored to a specific service provider.

Field Name	Field Type	Description
`formula`	string	The value of the attribute.
`key`	string	The attribute's identifier.

ConnectedAppCanvasConfig

The configuration options of the connected app if it's exposed as a canvas app.

Field Name	Field Type	Description
`accessMethod`	AccessMethod (enumeration of type string)	Indicates how the canvas app initiates the OAuth authentication flow. The valid values are: • `Get`—OAuth authentication is used, and the user is prompted to allow the third-party application to access their information. When you use this access method, the canvas app must initiate the OAuth authentication flow. • `Post`—OAuth authentication is used, but when the administrator installs the canvas app, they implicitly allow access for users. Therefore, the user won't be prompted to allow the third-party to access their user information. When you use this access method, the authentication is posted directly to the canvas app URL.
`canvasUrl`	string	The URL of the third-party app that's exposed as a canvas app.
`lifecycleClass`	string	The name of the `Canvas.CanvasLifecycleHandler` Apex class, if you've implemented this class for custom parameters. This field is available in API version 31.0 and later.
`locations`	CanvasLocationOptions (enumeration of type string)	Indicates where the canvas app can appear to the user. The valid values are: • `Aura`—Reserved for future use. • `AppLauncher`—Reserved for future use. • `Chatter`—The canvas app can appear in the app navigation list on the Chatter tab. • `ChatterFeed`—The canvas app can appear as a Chatter feed item. • `MobileNav`—The canvas app can appear in a mobile card in the Salesforce1 app. This value is available in API version 31.0 and later. • `None`—The canvas app can appear only in the Canvas App Previewer. • `OpenCTI`—The canvas app can appear in the call control tool. • `PageLayout`—The canvas app can appear on a page layout. When viewed in the Salesforce1 app, the canvas app appears in the record detail page. This value is available in API version 31.0 and later. • `Publisher`—The canvas app can appear as a global action. • `ServiceDesk`—The canvas app can appear in the footer or sidebars of a Salesforce console. • `UserProfile`—Reserved for future use.

Field Name	Field Type	Description
		• `Visualforce`—The canvas app can appear on a Visualforce page.
`options`	CanvasOptions (enumeration of type string)	Indicates whether you want to hide the share button and header in the publisher for your canvas app, and whether the app is a canvas personal app. Valid values are: • `HideShare`—The Share button is hidden in the publisher for the related canvas app. Available in API version 30.0 and later. • `HideHeader`—The header is hidden in the publisher for the related canvas app. Available in API version 30.0 and later. • `PersonalEnabled`—The app can be installed by end users as a canvas personal app. Available in API version 32.0 and later.
`samlInitiationMethod`	SamlInitiationMethod (enumeration of type string)	If you're using SAML single sign-on (SSO), indicates which provider initiates the SSO flow. • `IdpInitiated`—Identity provider initiated. Salesforce makes the initial request to start the SSO flow. • `SpInitiated`—Service provider initiated. The canvas app starts the SSO flow after it's invoked. • `None`—The canvas app isn't using SAML SSO. This field is available in API version 31.0 and later.

ConnectedAppIpRange

The list of IP addresses that can access the app without requiring the user to authenticate.

Field Name	Field Type	Description
`description`	string	Use this field to identify the purpose of the range, such as which part of a network corresponds to this range. This field is available in API version 31.0 and later.
`startAddress`	string	The first address in the IP range, inclusive.
`endAddress`	string	The last address in the IP range, inclusive.

ConnectedAppOauthConfig

Represents the field names that make up a custom attribute in a ConnectedApp.

Field Name	Field Type	Description
`callbackUrl`	string	The endpoint that Salesforce calls back to your application during OAuth; it's the OAuth `redirect_uri`.
`certificate`	string	The PEM-encoded certificate string, if the app uses a certificate.
`consumerKey`	string	A value used by the consumer for identification to Salesforce. Referred to as `client_id` in OAuth 2.0. In API version 32.0 and later, this field write-enabled. Once set, the value cannot be edited. The value must be alphanumeric (no special characters and no spaces) and a minimum of eight characters (maximum of 256 characters). If you specify a key already in use for another connected app in the organization, you'll get an error.
`consumerSecret`	string	A value that is combined with the `consumerKey` and used by the consumer for identification to Salesforce. Referred to as `client_secret` in OAuth 2.0. Typically, this value is generated by Salesforce when you create the connected app. However, the field is write-enabled so you can customize the shared secret value. Once set, the value is not returned in metadata API requests. The value must be alphanumeric (no special characters and no spaces) and a minimum of eight characters (maximum of 256 characters). If you specify a secret already in use for another connected app in the organization, you'll get an error. This field is available in API version 32.0 and later.
`scopes`	ConnectedAppOauthAccessScope (enumeration of type string)	The scopes refer to permissions given by the user running the connected app. When deploying metadata, valid values are: • `Address`—Allows access to the logged-in user's street address (the same behavior as deploying `Basic`). • `Api`—Allows access to the logged-in user's account over the APIs. • `Basic`—Allows access to your identity URL service (the same behavior as deploying `Address`, `Email`, `Phone`, and `Profile`). • `Chatter`—Allows access to only the Chatter REST API resources. • `CustomApplications`—Provides access to custom applications, such as those using Visualforce. • `CustomPermissions`—Allows access to the custom permissions in an organization associated with the connected app, and shows whether the current user has each permission enabled. • `Email`—Allows access to the logged-in user's email address (the same behavior as deploying `Basic`). • `Full`—Allows access to all data accessible by the logged-in user. • `OfflineAccess`—Allows the app to interact with the user's data while the user is offline, and get a refresh token (the same behavior as deploying `RefreshToken`).

Field Name	Field Type	Description
		• `OpenID`—Allows access to the logged-in user's unique identifier for OpenID Connect apps. • `Phone`—Allows access to the logged-in user's phone number value (the same behavior as deploying `Basic`). • `Profile`—Allows access to the logged-in user's profile (the same behavior as deploying `Basic`). • `RefreshToken`—Allows a refresh token to be returned if you're eligible to receive one (the same behavior as deploying `OfflineAccess`). • `Wave`—Allows access to the Wave REST API resources. Available in API version 35.0 and later. • `Web`—Allows the ability to use the `access_token` on the Web. This also includes `visualforce`, allowing access to Visualforce pages. When retrieving metadata, valid values are: • `Api`—Allows access to the logged-in user's account over the APIs. • `Basic`—Allows access to the user's identity URL service, and includes `Address`, `Email`, `Phone`, and `Profile`. • `Chatter`—Allows access to only the Chatter REST API resources. • `CustomApplications`—Allows access to custom applications, such as those using Visualforce. • `Full`—Allows access to all data accessible by the logged-in user. • `OpenID`—Allows access to the logged in user's unique identifier for OpenID Connect apps. • `RefreshToken`—Allows a refresh token to be returned if you are eligible to receive one, and is synonymous with allowing `OfflineAccess`. • `Wave`—Allows access to the Wave REST API resources. Available in API version 35.0 and later. • `Web`—Allows the ability to use the `access_token` on the Web. This also includes `visualforce`, allowing access to Visualforce pages.

ConnectedAppSamlConfig

Specifies how an app uses single sign-on.

Field Name	Field Type	Description
`acsUrl`	string	The assertion consumer service URL from the service provider.
`certificate`	string	The PEM-encoded certificate string, if the app uses a certificate.

Field Name	Field Type	Description
`entityUrl`	string	The entity ID from your service provider.
`encryptionCertificate`	string	Note: This information applies to Classic Encryption and not to Platform Encryption. The name of the certificate to use for encrypting SAML assertions to the service provider. This certificate is saved in the organization's Certificate and Key Management list. Available in API version 30.0 and later .
`encryptionType`	SamlEncryptionType (enumeration of type string)	Note: This information applies to Classic Encryption and not to Platform Encryption. When Salesforce is the identity provider, the SAML configuration can specify the encryption method used for encrypting SAML assertions to the service provider. The service provider detects the encryption method in the SAML assertion for decryption. Valid values are: • `AES_128`—128–bit key. • `AES_256`—256–bit key. • `Triple_Des`—Triple Data Encryption Algorithm. Available in API version 30.0 and later.
`issuer`	string	A URI that sends the SAML response. It can be used by the service provider to determine which identity provider sent the response. Available in API version 29.0 and later.
`samlNameIdFormat`	SamlNameIdFormatType (enumeration of type string)	Indicates the format the service provider (SP) requires for the user's single sign-on identifier. Available in API version 29.0 and later. Valid values are: • `Unspecified`—No format given. This is the default. • `EmailAddress`—Used if the subject type is the user's name ora federation ID (an ID internal to the SP). • `Persistent`—Used with the user ID and persistent ID subject types. • `Transient`—Used when the subject type is a custom attribute and can change every time the user logs in.
`samlSubjectCustomAttr`	string	If the `samlSubjectType` is `CustomAttr`, include that custom value here; otherwise, leave empty. Available in API version 29.0 and later.
`samlSubjectType`	SamlSubjectType (enumeration of type string)	The single sign-on identifier for the user. Valid values are: • `Username`—The user's Salesforce name. • `FederationId`—The user's identifier at the service provider. Get this value from the service provider. • `UserId`—The user's Salesforce identifier.

Field Name	Field Type	Description
		• `PersistentID`—A persistent opaque identifier that is specific to the identity provider and a service provider. • `CustomAttr`—The identifier is taken from a custom field value in `samlSubjectCustomAttr`.

Declarative Metadata Sample Definition

The following is an example package manifest used to deploy or retrieve the ConnectedApp metadata for an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>PortalTestApp</members>
        <name>ConnectedApp</name>
    </types>
    <version>29.0</version>
</Package>
```

The following is an example of a ConnectedApp component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConnectedApp xmlns="http://soap.sforce.com/2006/04/metadata">
  <fullName>AConnectedApp</fullName>
  <attributes>
    <formula>$User.CompanyName</formula>
    <key>companyName</key>
  </attributes>
  <contactEmail>joe@company.com</contactEmail>
  <mobileStartUrl>https://m.connectedapp.company.com</mobileStartUrl>
  <label>A ConnectedApp</label>
  <oauthConfig>
    <callbackUrl>https://callback.company.com</callbackUrl>
    <scopes>Basic</scopes>
    <scopes>Chatter</scopes>
  </oauthConfig>
  <samlConfig>
    <acsUrl>http://acs.company.com</acsUrl>
    <entityUrl>http://samlentityId.company.com</entityUrl>
    <samlSubjectType>Username</samlSubjectType>
  </samlConfig>
  <startUrl>https://connectedapp.company.com</startUrl>
  <ipRanges>
    <endAddress>10.0.0.46</endAddress>
    <startAddress>10.0.0.42</startAddress>
  </ipRanges>
  <ipRanges>
    <endAddress>10.0.0.32</endAddress>
    <startAddress>10.0.0.25</startAddress>
  </ipRanges>
</ConnectedApp>
```

Usage

If you're constructing a SAML-enabled connected app using Metadata API, and need to set the `IdP-Initiated Login URL` for your service provider, you have two options:

You can use the service provider app ID with the `app` parameter in the following format. This value is displayed in the Salesforce user interface. From Setup, enter "Connected Apps" in the `Quick Find` box, then select **Connected Apps**, then click the name of the connected app to see its detail page.

```
https://<Salesforce_base_URL>/idp/login?app=<app_id>
```

Or, if you're configuring the connected app using Metadata API only, you can use the `apiName` parameter of the service provider app in the following format. The `apiName` parameter is the `fullName` inherited from the Metadata type.

```
https://<Salesforce_base_URL>/idp/login?apiName=<fullName>
```

CorsWhitelistOrigin

Represents an origin in the CORS whitelist.

File Suffix and Directory Location

CorsWhitelistOrigin components have the suffix `.corswhitelistorigin` and are stored in the `corswhitelistorigins` folder.

Version

CorsWhitelistOrigin components are available in API version 32.0 and later.

Fields

Field Name	Field Type	Description
`developerName`	String	A unique name for the entry.
`urlPattern`	String	A URL pattern for the origin. The origin URL pattern must include the HTTPS protocol and a domain name, and may include a port. The wildcard character (*) is supported and must be in front of a second-level domain name. For example, `https://*.example.com` adds all subdomains of `example.com` to the whitelist. The origin URL pattern can be an IP address. However, an IP address and a domain that resolve to the same address are not the same origin and you must add them to the CORS whitelist as separate entries.

Declarative Metadata Sample Definition

The following is an example package manifest used to deploy or retrieve the CorsWhitelistOrigin metadata for an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CorsWhitelistOrigin</name>
    </types>
    <version>32.0</version>
</Package>
```

The following is an example of a CorsWhitelistOrigin component.

```
<?xml version="1.0" encoding="UTF-8"?>
<CorsWhitelistOrigin xmlns="http://soap.sforce.com/2006/04/metadata">
    <developerName>CorsWhitelistEntry1</developerName>
    <urlPattern>https://*.example.com</urlPattern>
</CorsWhitelistOrigin>
```

Usage

CORS (cross-origin resource sharing) is a W3C recommendation that enables Web browsers to request resources from origins other than their own. For example, using CORS, a JavaScript script at `https://www.example.com` could request a resource from `https://www.salesforce.com`.

If a browser that supports CORS makes a request to an origin in the Salesforce CORS whitelist, Salesforce returns the origin in the `Access-Control-Allow-Origin` HTTP header, along with any additional CORS HTTP headers. If the origin is not whitelisted, Salesforce returns HTTP status code 404.

CustomApplication

CustomApplication represents a custom or standard application. In API version 29.0 and earlier, CustomApplication represents only a custom application. An application is a list of tab references, with a description and a logo. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Custom and standard applications have the suffix `.app` and are stored in the `applications` folder.

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 applications are available in API version 10.0 and later. Standard applications are available in API version 30.0 and later.

Fields

Field Name	Field Type	Description
customApplicationComponents	CustomApplicationComponents	Represents custom console components (Visualforce pages) assigned to a Salesforce console app.
defaultLandingTab	string	The fullName of a standard tab or custom tab that opens when this application is selected.
description	string	The optional description text of the application.
detailPageRefreshMethod	string	Determines how detail pages refresh in a Salesforce console app. Required if isServiceCloudConsole is true. The valid values are: • none • autoRefresh • flag This field is available in API version 25.0 and later.
domainWhitelist	DomainWhitelist	Any external domains that users can access from within a Salesforce console app. For example, www.yourdomain.com. This field is available in API version 25.0 and later.
enableCustomizeMyTabs	boolean	Indicates if a Salesforce console app has Customize My Tabs enabled. If enabled, users can hide, display, and organize items in the navigation tab. This field is available in API version 34.0 and later.
enableKeyboardShortcuts	boolean	Indicates if a Salesforce console app has keyboard shortcuts enabled, which let users perform actions by pressing a combination of keys instead of having to use a mouse. After keyboard shortcuts are enabled, several default shortcuts are available for customization. Before you can create custom shortcuts, a developer must define the shortcut's action with the addEventListener() method in the Salesforce Console Integration Toolkit. You can't create keyboard shortcuts for actions performed outside of the console. This field is required if isServiceCloudConsole is true. This field is available in API version 28.0 and later.
enableMultiMonitorComponents	boolean	Indicates if a Salesforce console app has multi-monitor components enabled, which lets users move portions of a console from their browsers to locations on their

Field Name	Field Type	Description
		screens. This field is required if `isServiceCloudConsole` is `true`. This field is available in API version 30.0 and later.
`enablePinTabs`	boolean	Indicates if a Salesforce console app has pinned tabs enabled, which lets users pin primary tabs to the tab bar for quick access. This field is available in API version 34.0 and later.
`enableTabHover`	boolean	Indicates if a Salesforce console app has tab hover enabled, which displays summary information about a record in an overlay when the user hovers over a tab. This field is available in API version 36.0 and later.
`enableTabLimits`	boolean	Indicates whether limits are enabled on the number of primary tabs and subtabs that can be opened in a Salesforce console session. When `true`, values for `tabLimitConfig` are required This field is available in API version 36.0 and later.
`footerColor`	string	Determines the footer color in a Salesforce console app. Specify the color with a hexadecimal code, such as #0000FF for blue.
`headerColor`	string	Determines the header color in a Salesforce console app. Specify the color with a hexadecimal code, such as #0000FF for blue.
`isServiceCloudConsole`	boolean	Indicates if the application is a Salesforce console app. For more information, see "Salesforce Console" in the Salesforce online help.
`keyboardShortcuts`	KeyboardShortcuts	Represents the keyboard shortcuts for a Salesforce console app. Keyboard shortcuts let users perform actions by pressing a combination of keys instead of having to use a mouse. This field is available in API version 28.0 and later.
`fullName`	string	The internal name of the application, based on the `label`, but with white spaces and special characters escaped out for validity. The `fullName` 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.
`label`	string	The name of the application.

Field Name	Field Type	Description
`listPlacement`	ListPlacement	Represents how lists display in a Salesforce console app. Required if `isServiceCloudConsole` is `true`.
`listRefreshMethod`	string	Determines how lists refresh in a Salesforce console app. Required if `isServiceCloudConsole` is `true`. The valid values are: • none • refreshList • refreshListRows This field is available in API version 25.0 and later.
`liveAgentConfig`	LiveAgentConfig on page 189	Represents the configurations for using Live Agent in the Salesforce Console.
`logo`	string	The optional reference to the image document for the application.
`primaryTabColor`	string	Determines the primary tab color in a Salesforce console app. Specify the color with a hexadecimal code, such as #0000FF for blue.
`pushNotifications`	PushNotifications	Represents push notifications for a Salesforce console app. Push notifications are visual indicators on lists and detail pages that show when a record or field has changed during a user's session. For example, assume that two support agents are working on the same case. If one agent changes the `Priority`, a push notification displays to the other agent so the agent notices the change and doesn't duplicate the effort. This field is available in API version 28.0 and later.
`saveUserSessions`	boolean	Indicates if a Salesforce console app saves user sessions automatically. If enabled, when console users close their browsers or log out of Salesforce, any previously open tabs display when users log in again. Required if `isServiceCloudConsole` is `true`. This field is available in API version 28.0 and later.
`tab`	string	The list of tabs included in this application. In API version 12.0, the `fullName` for built-in tabs like Home, Account, and Reports, is the name of the tab (Home, for example). In API version 13.0 and later, built-in tabs are prefixed with `standard-`. For example, to reference the Account tab you would use `standard-Account`.

Field Name	Field Type	Description
`tabLimitConfig`	TabLimitConfig	Represents the maximum number of primary tabs and subtabs allowed in one Salesforce console session. Required if `enableTabLimits` is `true`. This field is available in API version 36.0 and later.
`workspaceMappings`	WorkspaceMappings	Represents how records open in a Salesforce console app. Required if `isServiceCloudConsole` is `true`. This field is available in API version 25.0 and later.

CustomApplicationComponents

Represents custom console components (Visualforce pages) assigned to a Salesforce console app. Available in API version 25.0 and later.

Field Name	Field Type	Description
`alignment`	string	Determines how custom console components are aligned in the footer of a Salesforce console app.
`customApplicationComponent`	string	The name of a custom console component assigned to a Salesforce console app.

CustomShortcut

Represents custom keyboard shortcuts assigned to a Salesforce console app. Before you can create custom shortcuts, a developer must define the shortcut's action with the `addEventListener()` method in the Salesforce Console Integration Toolkit. You can't create keyboard shortcuts for actions performed outside of the console. Available in API version 28.0 and later.

Field Name	Field Type	Description
`action`	string	Required. The action performed in the console when a user presses the keyboard shortcut.
`active`	boolean	Required. Indicates whether the keyboard shortcut is active (`true`) or not (`false`).
`keyCommand`	string	Required. The combination of keys a user presses to trigger the keyboard shortcut. Keyboard shortcuts aren't case-sensitive, but they display as uppercase on setup pages in the Salesforce user interface so that they're easier to read. Each key command can include up to four modifier keys followed by one non-modifier key. Modifier and non-modifier keys are separated by the `+` key. Modifier keys can occur in any order, but you must place non-modifier keys at the end of the key command sequence. For example, `SHIFT+CTRL+ALT+META +A`. Valid modifier keys are: • `SHIFT`

Field Name	Field Type	Description
		• CTRL • ALT • META (represents the COMMAND key on Macs) Valid non-modifier keys are letters A through Z and numbers 0 through 9. Other valid keys are: • TAB • ENTER • PAUSE/BREAK • CAPS LOCK • ESC • SPACE • PAGE UP • PAGE DOWN • END • HOME • LEFT ARROW • UP ARROW • RIGHT ARROW • DOWN ARROW • PRINT SCREEN • INSERT • DELETE • RIGHT WINDOW • NUMPAD 0 • NUMPAD 1 • NUMPAD 2 • NUMPAD 3 • NUMPAD 4 • NUMPAD 5 • NUMPAD 6 • NUMPAD 7 • NUMPAD 8 • NUMPAD 9 • MULTIPLY • ADD • SUBTRACT • DECIMAL POINT • DIVIDE

Field Name	Field Type	Description
		• F1 • F2 • F3 • F4 • F5 • F6 • F7 • F8 • F9 • F10 • F11 • F12 • NUM LOCK • SCROLL LOCK • ; • = • , • – • . • / • ` • [•] • \ • '
`description`	string	The optional description text for the keyboard shortcut.
`eventName`	string	Required. Code available to developers who want to add custom shortcut functions to the console via the Salesforce Console Integration Toolkit.

DefaultShortcut

Represents default keyboard shortcuts assigned to a Salesforce console app. Once you enable keyboard shortcuts for a console, several default shortcuts are available for customization, such as opening and closing tabs, moving between tabs, and saving records. Available in API version 28.0 and later.

Field Name	Field Type	Description
`action`	string	Required. The action performed in the console when a user presses the keyboard shortcut. Valid values are: • `FOCUS_CONSOLE` • `FOCUS_NAVIGATOR_TAB` • `FOCUS_DETAIL_VIEW` • `FOCUS_PRIMARY_TAB_PANEL` • `FOCUS_SUBTAB_PANEL` • `FOCUS_LIST_VIEW` • `FOCUS_FIRST_LIST_VIEW` • `FOCUS_SEARCH_INPUT` • `MOVE_LEFT` • `MOVE_RIGHT` • `UP_ARROW` • `DOWN_ARROW` • `OPEN_TAB_SCROLLER_MENU` • `OPEN_TAB` • `CLOSE_TAB` • `ENTER` • `EDIT` • `SAVE` For a list and description of the default keyboard shortcuts, see "Default Keyboard Shortcuts for a Salesforce Console" in the Salesforce online help.
`active`	boolean	Required. Indicates whether the keyboard shortcut is active (`true`) or not (`false`).
`keyCommand`	string	Required. The combination of keys a user presses to trigger the keyboard shortcut. Keyboard shortcuts aren't case-sensitive, but they display as uppercase on setup pages in the Salesforce user interface so that they're easier to read. Each key command can include up to four modifier keys followed by one non-modifier key. Modifier and non-modifier keys are separated by the `+` key. Modifier keys can occur in any order, but you must place non-modifier keys at the end of the key command sequence. For example, `SHIFT+CTRL+ALT+META +A`. Valid modifier keys are: • `SHIFT` • `CTRL` • `ALT` • `META` (represents the COMMAND key on Macs)

Field Name	Field Type	Description
		Valid non-modifier keys are letters A through Z and numbers 0 through 9. Other valid keys are: • TAB • ENTER • PAUSE/BREAK • CAPS LOCK • ESC • SPACE • PAGE UP • PAGE DOWN • END • HOME • LEFT ARROW • UP ARROW • RIGHT ARROW • DOWN ARROW • PRINT SCREEN • INSERT • DELETE • RIGHT WINDOW • NUMPAD 0 • NUMPAD 1 • NUMPAD 2 • NUMPAD 3 • NUMPAD 4 • NUMPAD 5 • NUMPAD 6 • NUMPAD 7 • NUMPAD 8 • NUMPAD 9 • MULTIPLY • ADD • SUBTRACT • DECIMAL POINT • DIVIDE • F1 • F2 • F3

Field Name	Field Type	Description
		• F4 • F5 • F6 • F7 • F8 • F9 • F10 • F11 • F12 • NUM LOCK • SCROLL LOCK • ; • = • , • – • . • / • ` • [•] • \ • '

DomainWhitelist

Represents any external domains that users can access from within a Salesforce console app. For example, `www.yourdomain.com`. Available in API version 25.0 and later.

Field Name	Field Type	Description
`domain`	string	The external domains that users can access from within this Salesforce console app.

KeyboardShortcuts

Represents keyboard shortcuts assigned to a Salesforce console app. Required if `isServiceCloudConsole` is `true`. Available in API version 28.0 and later.

Field Name	Field Type	Description
`customShortcut`	KeyboardShortcuts[]	Represents custom keyboard shortcuts assigned to a Salesforce console app. Before you can create custom shortcuts, a developer must define the shortcut's action with the `addEventListener()` method in the Salesforce Console Integration Toolkit. You can't create keyboard shortcuts for actions performed outside of the console.
`defaultShortcut`	KeyboardShortcuts[]	Represents default keyboard shortcuts assigned to a Salesforce console app. Once you enable keyboard shortcuts for a console, several default shortcuts are available for customization, such as opening and closing tabs, moving between tabs, and saving records. For a list and description of the default keyboard shortcuts, see "Default Keyboard Shortcuts for a Salesforce Console" in the Salesforce online help.

ListPlacement

Represents how lists display in a Salesforce console app. Required if `isServiceCloudConsole` is `true`. Available in API version 25.0 and later.

Field Name	Field Type	Description
`height`	int	Height of the list in pixels or percentage. Required if `location` is top.
`location`	string	Required. Location of the list on the screen. Valid values are: • full • top • left
`units`	string	Required. Represents if `height` or `width` is in pixels or percentage.
`width`	int	Width of the list in pixels or percentage. Required if `location` is left.

LiveAgentConfig

Represents your organization's settings for using Live Agent in the Salesforce Console.

Field Name	Field Type	Description
`enableLiveChat`	boolean	Specifies whether Live Agent is enabled in your organization (`true`) or not (`false`).
`openNewAccountSubtab`	boolean	Specifies whether to open a new Account subtab in the Salesforce Console automatically (`true`) or not (`false`) when an agent accepts a chat.
`openNewCaseSubtab`	boolean	Specifies whether to open a new Case subtab in the Salesforce Console automatically (`true`) or not (`false`) when an agent accepts a chat.

Field Name	Field Type	Description
`openNewContactSubtab`	boolean	Specifies whether to open a new Contact subtab in the Salesforce Console automatically (`true`) or not (`false`) when an agent accepts a chat.
`openNewLeadSubtab`	boolean	Specifies whether to open a new Lead subtab in the Salesforce Console automatically (`true`) or not (`false`) when an agent accepts a chat.
`openNewVFPageSubtab`	boolean	Specifies whether to open a new Visualforce page as a subtab in the Salesforce Console automatically (`true`) or not (`false`) when an agent accepts a chat.
`pagesToOpen`	PagesToOpen on page 190	Specifies the Visualforce pages to open in subtabs when an agent accepts a chat in the Salesforce Console.
`showKnowledgeArticles`	boolean	Specifies whether to display the Knowledge component while using Live Agent in the Salesforce Console (`true`) or not (`false`).

PagesToOpen

Represents the Visualforce pages you want to open in subtabs when an agent accepts a chat request in the Salesforce Console. Available in API version 28.0 and later.

Field Name	Field Type	Description
`pagesToOpen`	string	The name of the Visualforce pages you want to open in subtabs when an agent accepts a chat in the Salesforce Console.

PushNotifications

Represents a set of push notifications, which are visual indicators on lists and detail pages that show when a record or field has changed during a user's session. Available for use if `isServiceCloudConsole` is `true`. Available in API version 28.0 and later.

Field Name	Field Type	Description
`pushNotification`	PushNotification[]	The set of push notifications.

PushNotification

Represents if visual indicators on lists and detail pages display in a Salesforce console app when a record or field has changed during a user's session. Available for use if `isServiceCloudConsole` is `true`. Available in API version 28.0 and later.

Field Name	Field Type	Description
`fieldNames`	string	Required. The name of the field, or fields, that trigger push notifications for the selected object.
`objectName`	string	Required. Name of the object that triggers push notifications.

TabLimitConfig

Represents the maximum number of primary tabs and subtabs allowed in one Salesforce console session. Required if `enableTabLimits` is `true`. Available in API version 36.0 and later.

Field Name	Field Type	Description
`maxNumberOfPrimaryTabs`	string	The maximum number of primary tabs allowed in one console session. Valid values are: • 5 • 10 • 20 • 30
`maxNumberOfSubTabs`	string	The maximum number of subtabs allowed in one console session. Valid values are: • 5 • 10 • 15

WorkspaceMappings

Represents how records open in a Salesforce console app. Required if `isServiceCloudConsole` is `true`. Available in API version 25.0 and later.

Field Name	Field Type	Description
`mapping`	WorkspaceMapping	Represents how records for a specific tab open in a Salesforce console app. Required for each tab specified in the CustomApplication.

WorkspaceMapping

Represents how records for a specific tab open in a Salesforce console app. Required for each tab specified in the CustomApplication. Available in API version 25.0 and later.

Field Name	Field Type	Description
`fieldName`	string	The name of the field that specifies the primary tab in which to display `tab` as a subtab. If not specified, `tab` opens as a primary tab.
`tab`	string	Required. Name of the tab.

Retrieving Apps

To retrieve apps in your organization, use the CustomApplication type name in the `package.xml` manifest file. You can either retrieve all apps or specify which apps to retrieve in the types section of `package.xml`.

To retrieve all apps in your organization—custom and standard apps, specify the wildcard character (*), as follows.

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

Note: In API version 29.0 and earlier, use of the wildcard returns only all custom applications but not standard applications.

To retrieve a custom app, specify the app name.

```
<types>
    <members>MyCustomApp</members>
    <name>CustomApplication</name>
</types>
```

To retrieve a standard app, add the `standard__` prefix to the app name. For example, to retrieve the Chatter standard app, specify `standard__Chatter`.

```
<types>
    <members>standard__Chatter</members>
    <name>CustomApplication</name>
</types>
```

To retrieve an app that is part of an installed package, add the package namespace prefix followed by two underscores and the app name. For example, if the package namespace is `myInstalledPackageNS` and the app name is `PackageApp`, specify `myInstalledPackageNS__PackageApp`, as follows.

```
<types>
    <members>myInstalledPackageNS__PackageApp</members>
    <name>CustomApplication</name>
</types>
```

Declarative Metadata Sample Definition

The following is the definition of a custom app:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
    <defaultLandingTab>Myriad_Publishing__c</defaultLandingTab>
    <description>App to manage Myriad Publishing</description>
    <label>Myriad</label>
    <logo>MyriadFolder/Myriad_Logo.jpg</logo>
    <tab>standard-Chatter</tab>
    <tab>standard-File</tab>
    <tab>Myriad_Publishing__c</tab>
    <tab>standard-report</tab>
    <tab>standard-Dashboard</tab>
</CustomApplication>
```

The following is a definition of a standard app (Chatter):

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
    <defaultLandingTab>standard-home</defaultLandingTab>
    <label>Collaboration</label>
```

```
    <tab>standard-Chatter</tab>
    <tab>standard-UserProfile</tab>
    <tab>standard-OtherUserProfile</tab>
    <tab>standard-CollaborationGroup</tab>
    <tab>standard-File</tab>
</CustomApplication>
```

Declarative Metadata Sample Definition—Salesforce Console

The following is the definition of a custom app where `isServiceCloudConsole` is `true`:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
    <customApplicationComponents>
       <alignment>left</alignment>
       <customApplicationComponent>MyComponent</customApplicationComponent>
    </customApplicationComponents>
    <defaultLandingTab>standard-home</defaultLandingTab>
    <detailPageRefreshMethod>autoRefresh</detailPageRefreshMethod>
    <isServiceCloudConsole>true</isServiceCloudConsole>
    <keyboardShortcuts>
       <customShortcut>
         <action>MyCustomShortcutAction</action>
         <active>true</active>
         <keyCommand>X</keyCommand>
         <description>Custom Shortcut example</description>
         <eventName>myCustomShortcutExample</eventName>
       </customShortcut>
       <defaultShortcut>
         <action>FOCUS_CONSOLE</action>
         <active>true</active>
         <keyCommand>ESC</keyCommand>
       </defaultShortcut>
       <defaultShortcut>
         <action>FOCUS_NAVIGATOR_TAB</action>
         <active>true</active>
         <keyCommand>V</keyCommand>
       </defaultShortcut>
       <defaultShortcut>
         <action>FOCUS_DETAIL_VIEW</action>
         <active>true</active>
         <keyCommand>SHIFT+S</keyCommand>
       </defaultShortcut>
       <defaultShortcut>
         <action>FOCUS_PRIMARY_TAB_PANEL</action>
         <active>true</active>
         <keyCommand>P</keyCommand>
       </defaultShortcut>
       <defaultShortcut>
         <action>FOCUS_SUBTAB_PANEL</action>
         <active>true</active>
         <keyCommand>S</keyCommand>
       </defaultShortcut>
       <defaultShortcut>
```

```
    <action>FOCUS_LIST_VIEW</action>
    <active>true</active>
    <keyCommand>N</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>FOCUS_FIRST_LIST_VIEW</action>
    <active>true</active>
    <keyCommand>SHIFT+F</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>FOCUS_SEARCH_INPUT</action>
    <active>true</active>
    <keyCommand>R</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>MOVE_LEFT</action>
    <active>true</active>
    <keyCommand>LEFT ARROW</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>MOVE_RIGHT</action>
    <active>true</active>
    <keyCommand>RIGHT ARROW</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>UP_ARROW</action>
    <active>true</active>
    <keyCommand>UP ARROW</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>DOWN_ARROW</action>
    <active>true</active>
    <keyCommand>DOWN ARROW</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>OPEN_TAB_SCROLLER_MENU</action>
    <active>true</active>
    <keyCommand>D</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>OPEN_TAB</action>
    <active>true</active>
    <keyCommand>T</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>CLOSE_TAB</action>
    <active>true</active>
    <keyCommand>C</keyCommand>
  </defaultShortcut>
  <defaultShortcut>
    <action>ENTER</action>
    <active>true</active>
    <keyCommand>ENTER</keyCommand>
  </defaultShortcut>
```

```
        <defaultShortcut>
          <action>EDIT</action>
          <active>true</active>
          <keyCommand>E</keyCommand>
        </defaultShortcut>
        <defaultShortcut>
          <action>SAVE</action>
          <active>true</active>
          <keyCommand>CTRL+S</keyCommand>
        </defaultShortcut>
    </keyboardShortcuts>
    <label>MyConsole</label>
    <listPlacement>
        <location>left</location>
        <units>percent</units>
        <width>20</width>
    </listPlacement>
    <listRefreshMethod>refreshList</listRefreshMethod>
    <pushNotifications>
        <pushNotification>
          <fieldNames>CreatedBy</fieldNames>
          <objectName>Campaign</objectName>
        </pushNotification>
        <pushNotification>
          <fieldNames>CustomField1__c</fieldNames>
          <objectName>CustomObject1__c</objectName>
        </pushNotification>
    </pushNotifications>
    <saveUserSessions>false</saveUserSessions>
    <tab>standard-Case</tab>
    <tab>standard-Account</tab>
    <tab>standard-Contact</tab>
     <tab>standard-Contract</tab>
    <workspaceMappings>
        <mapping>
           <tab>standard-Case</tab>
        </mapping>
        <mapping>
           <fieldName>AccountId</fieldName>
           <tab>standard-Contract</tab>
        </mapping>
        <mapping>
           <tab>standard-Contract</tab>
        </mapping>
        <mapping>
           <fieldName>ParentId</fieldName>
           <tab>standard-Account</tab>
        </mapping>
    </workspaceMappings>
</CustomApplication>
```

SEE ALSO:

CustomTab

CustomApplicationComponent

Represents a custom console component (Visualforce page) assigned to a CustomApplication that is marked as a Salesforce console. Custom console components extend the capabilities of Salesforce console apps. See "Console Components" in the Salesforce online help.

File Suffix and Directory Location

Custom application components have the suffix `.customApplicationComponent` and are stored in the `customApplicationComponents` folder.

Version

Custom applications are available in API version 25.0 and later.

Fields

Field Name	Field Type	Description
`buttonIconUrl`	string	The address of a page that hosts an icon for the button.
`buttonStyle`	string	The inline style used to define how the button looks.
`buttonText`	string	The label on the button used to launch the custom console component.
`buttonWidth`	int	The pixel width of the button as it should display in the Salesforce console.
`height`	int	The pixel height of the window used to display the custom console component.
`isHeightFixed`	boolean	Required. Indicates whether users can change the custom console component height (`false`) or not (`true`).
`isHidden`	boolean	Required. Indicates whether the custom console component is hidden from users (`true`) or not (`false`).
`isWidthFixed`	boolean	Required. Indicates whether users can change the component width (`false`) or not (`true`).
`visualforcePage`	string	Required. Name of the Visualforce page that represents the custom console component.
`width`	int	The pixel width of the window used to display the custom console component.

Declarative Metadata Sample Definition

The following is the definition of a custom application component:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplicationComponent xmlns="http://soap.sforce.com/2006/04/metadata">
    <buttonIconUrl>http://www.salesforce.com</buttonIconUrl>
    <buttonStyle>buttonStyleCSS</buttonStyle>
    <buttonText>buttonText</buttonText>
    <buttonWidth>200</buttonWidth>
    <height>200</height>
    <isHeightFixed>false</isHeightFixed>
    <isHidden>false</isHidden>
    <isWidthFixed>false</isWidthFixed>
    <visualforcePage>MyVisualforcePage</visualforcePage>
    <width>50</width>
</CustomApplicationComponent>
```

CustomFeedFilter

Represents a custom feed filter that limits the feed view to feeds from the Cases object. The custom feed filter shows only feed items that satisfy the criteria specified in the CustomFeedFilter definition.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CustomFeedFilter components have the suffix `.feedFilter` and are stored in the `feedFilters` folder.

Version

CustomFeedFilter components are available in API version 35.0 and later.

Fields

Field Name	Field Type	Description
`description`	string	The description of the custom feed filter. For example, specify what feed items that filter shows.
`criteria`	FeedFilterCriterion []	The criterion that defines which feed items are shown when the filter is applied. The feed filter displays all feed items that satisfy the criteria.
`label`	string	Required. The API label of the custom feed filter.

FeedFilterCriterion

Represents the conditions that a feed item must satisfy to be displayed when a feed filter is applied.

Field Name	Field Type	Description
`feedItemType`	FeedItemType (enumeration of type string)	Required. The type of feed items that the filter shows. The feed item type can be one of the following values: • AttachArticleEvent • CallLogPost • CanvasPost • CaseCommentPost • ChangeStatusPost • ChatTranscriptPost • ContentPost • CreateRecordEvent • EmailMessageEvent • LinkPost • MilestoneEvent • QuestionPost • PollPost • ReplyPost • SocialPost • TextPost
`feedItemVisibility`	FeedItemVisibility (enumeration of type string)	The visibility of feed items that the filter shows. For example, you can show only poll posts that are visible internally. Valid values are: • AllUsers • InternalUsers
`relatedSObjectType`	string	The API name of the object that the feed item refers to. This field is typically used with the CreateRecordEvent feed item type. For example, a feed filter can show CreateRecordEvent feed items for the Cases object.

Declarative Metadata Sample Definition

The following is an example of a CustomFeedFilter component.

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

```
        <relatedSObjectType>MyCO01__c</relatedSObjectType>
    </criteria>
    <criteria>
        <feedItemType>CreateRecordEvent</feedItemType>
        <relatedSObjectType>Case</relatedSObjectType>
    </criteria>
    <criteria>
        <feedItemType>PollPost</feedItemType>
        <feedItemVisibility>InternalUsers</feedItemVisibility>
    </criteria>
    <label>Sample Custom Feed Filter</label>
</CustomFeedFilter>
```

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>myCaseFeedFilter</members>
        <name>CustomFeedFilter</name>
    </types>
    <version>36.0</version>
</Package>
```

CustomLabels

This metadata type allows you to create custom labels that can be localized for use in different languages, countries, and currencies. This type extends the Metadata metadata type and inherits its `fullName` field. Custom labels are custom text values, up to 1,000 characters in length, that can be accessed from Apex classes or Visualforce pages. For more information, see "Custom Labels" in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Master custom label values are stored in the `CustomLabels.labels` file. Translations are stored in a file with a name format of `Translation-`*`localeCode`*`.translation`, where *`localeCode`* is the locale code of the translation language. The supported locale codes are listed in Language on page 607.

Custom label translations are stored in the `labels` folder in the corresponding package directory.

Version

CustomLabels components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`fullName`	string	Required. The name of the custom label bundle. 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.
`labels`	CustomLabel[]	A list of custom labels.

CustomLabel

This metadata type represents a custom label. This type extends the Metadata metadata type and inherits its `fullName` field.

Field	Field Type	Description
`categories`	string	A comma-separated list of categories for the label. This field can be used in filter criteria when creating custom label list views. Maximum of 255 characters.
`fullName`	string	Required. The name of the custom label. 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.
`language`	string	Required. The language of the translated custom label.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`shortDescription`	string	Required. An easily recognizable term to identify this custom label. This description is used in merge fields.
`value`	string	Required. The translated custom label. Maximum of 1000 characters.

Usage

Use CustomLabels with the wildcard character (*) for members in the `package.xml` manifest file to retrieve all custom labels that are defined in your organization. CustomLabels doesn't support retrieving one or more custom labels by name. To retrieve specific labels by name, use CustomLabel and specify the label names as members.

Declarative Metadata Sample Definition

A sample XML definition of a custom label components is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomLabels xmlns="http://soap.sforce.com/2006/04/metadata">
    <labels>
        <fullName>quoteManual</fullName>
        <label>This is a manual quote.</label>
        <language>en_US</language>
        <protected>false</protected>
        <shortDescription>Manual Quote</shortDescription>
    </labels>
    <labels>
        <fullName>quoteAuto</fullName>
        <label>This is an automatically generated quote.</label>
        <language>en_US</language>
        <protected>false</protected>
        <shortDescription>Automatic Quote</shortDescription>
    </labels>
</CustomLabels>
```

This is a sample manifest file for retrieving all custom labels in the organization by using the CustomLabels type.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>MyPkg</fullName>
    <types>
      <members>*</members>
      <name>CustomLabels</name>
    </types>
    <version>36.0</version>
</Package>
```

This is a sample manifest file for retrieving two custom labels by name. Notice it uses the CustomLabel singular type.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>MyPkg</fullName>
    <types>
      <members>quoteManual</members>
      <members>quoteAuto</members>
      <name>CustomLabel</name>
    </types>
    <version>36.0</version>
</Package>
```

SEE ALSO:

Translations

Custom Metadata Types (CustomObject)

Represents the metadata associated with a custom metadata type.

For more information, see the *Custom Metadata Types Implementation Guide*.

File Suffix and Directory Location

A custom metadata type is defined as a custom object and is stored in the objects folder. Custom metadata types have a suffix of `__mdt` (instead of `__c` for custom objects). Custom metadata type field names have a suffix of `__c`, like other custom fields. Custom metadata type field names must be dot-qualified with the name of the custom metadata type to which they belong.

Names of custom metadata types must be unique within their namespace. All custom metadata types belong to the `CustomMetadata` namespace and can optionally belong to a second namespace. In your organization, you can use custom metadata types with your namespace as well as other organizations' namespaces.

Version

Custom metadata type components are available in API version 31.0 and later.

Special Access Rules

To create custom metadata types, you must have the "Author Apex" permission. Only managed package developers can add new fields to the custom metadata types in their managed packages. Customers who install a managed custom metadata type can't add new custom fields to it.

Fields

Custom metadata types can contain the following CustomObject fields.

To make the fields on your custom metadata types unique and indexable, mark your fields as `Unique` and `ExternalId`.

Field Name	Field Type	Description
`description`	string	A description of the custom metadata type. This field can contain a maximum of 1,000 characters.
`fields`	CustomField[]	Represents one or more custom fields in the custom metadata type.
`gender`	Gender	Indicates the gender of the noun that represents the object. This field is used for languages where words need different treatment depending on their gender.
`Protected`	boolean	When a custom metadata type is released in a managed package, access is limited in specific ways. • Code that's in the same managed package as custom metadata records can read the records. • Code that's in the same managed package as custom metadata types can read the records that belong to that type. • Code that's in a managed package that doesn't contain either the type or the protected record can't read the protected records.

Field Name	Field Type	Description
		• Code that the subscriber creates and code that's in an unmanaged package can't read the protected records. • The developer can modify protected records only with a package upgrade. The subscriber can't read or modify protected records. The developer name of a protected record can't be changed after release. • The subscriber can't create records of a protected type. Records that are hidden by these access rules are also unavailable to REST, SOAP, SOQL, and Setup.
`label`	string	A label that represents the object throughout the Salesforce Setup user interface. Custom metadata types are visible only through the recently used objects list on the Force.com Home Page and in the packaging user interface.
`pluralLabel`	string	The plural version of the label value.
`startsWith`	StartsWith (enumeration of type string)	Indicates whether the noun starts with a vowel, a consonant, or a special character. This field is used for languages where words need different treatment depending on their first character.
`visibility`	SetupObjectVisibility (enumeration of type string)	When this field is present, this component is not a custom object, but a custom setting or custom metadata type. This field returns the visibility of the custom setting or custom metadata type. The following values are valid. • `Public`—If the custom setting or custom metadata type is packaged, it's accessible to all subscribing organizations. • `Protected`—If the custom setting or custom metadata type is in a managed package, it's only accessible to the developer organization; subscribing organizations can't access it. The default value is `Public`.

Declarative Metadata Sample Definition

The sample application is based on a collaborative effort between two fictional organizations. Picklists R Us develops reusable enhancements to the Salesforce App Cloud that involve picklist-related functionality. TravelApp, Inc. develops an interplanetary travel application that uses picklist features from Picklists R Us. Galactic Tours is a customer of these organizations. Galactic Tours installs Picklists R Us's package and TravelApp, Inc.'s extension into its organization.

In this example, Picklists R Us creates its Reusable Picklist custom metadata type by deploying a file in the objects folder, named `ReusablePicklistOption__mdt.object`, with these contents.

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

```
   <fields>
      <fullName>AlphaSort__c</fullName>
      <defaultValue>false</defaultValue>
      <externalId>false</externalId>
      <label>Sorted Alphabetically</label>
      <type>Checkbox</type>
   </fields>
   <label>Reusable Picklist</label>
   <pluralLabel>Reusable Picklist</pluralLabel>
   <visibility>Public</visibility>
</CustomObject>
```

This excerpt from Picklists R Us's `package.xml` file shows the use of dot notation and the `__mdt` suffix. The `CustomMetadata` and Picklists R Us namespaces aren't shown here because these namespaces are common to all items in `package.xml`. If Picklists R Us used the namespace `picklist1234`, the full name of `ReusablePicklistOption__mdt` would be `picklist1234__ReusablePicklistOption__mdt`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
...
   <types>
      <members>PicklistTest__c.PicklistTestField__c</members>
      <members>ReusablePicklistOption__mdt.Picklist__c</members>
      <members>ReusablePicklistOption__mdt.SortOrder__c</members>
      <members>PicklistUsage__mdt.Field__c</members>
      <members>PicklistUsage__mdt.Picklist__c</members>
      <members>PicklistUsage__mdt.SObjectType__c</members>
      <members>ReusablePicklist__mdt.AlphaSort__c</members>
      <name>CustomField</name>
   </types>
...
   <types>
      <members>PicklistTest__c</members>
      <members>ReusablePicklistOption__mdt</members>
      <members>PicklistUsage__mdt</members>
      <members>ReusablePicklist__mdt</members>
      <name>CustomObject</name>
   </types>
...
   <version>36.0</version>
</Package>
```

CustomMetadata

Represents a record of a custom metadata type.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CustomMetadata components have the suffix `.md` and are stored in the `customMetadata` folder. Unlike custom metadata types, custom metadata records don't have a double-underscore suffix. Custom metadata record names are prepended with their custom metadata type name, excluding the `__mdt` suffix but including the namespace of any types in an installed managed package.

Version

CustomMetadata components are available in API version 31.0 and later.

Special Access Rules

To create custom metadata records, you must have the "Customize Application" permission.

Fields

Field Name	Field Type	Description
`description`	string	A description of the custom metadata record. This field can contain a maximum of 1,000 characters.
`label`	string	A label that represents the object throughout the Salesforce Setup user interface. Custom metadata records are currently visible only through the packaging user interface.
`values`	CustomMetadataValue[]	Represents one or more values for custom fields on the custom metadata record.

CustomMetadataValue

Represents a value for a custom field on the custom metadata record.

Field Name	Field Type	Description
`field`	string	Required. The non-object-qualified name of a custom field in the custom metadata type. This value corresponds to the name of a field on the custom metadata record's custom metadata type. Include the namespace (if the type is from a managed package) and the `__c` suffix. The name of the custom metadata type isn't required. For example, `picklist1234__AlphaSort__c`.
`value`	Any type	Required. The value for a field on the CustomMetadata object. This value can be null. For more information, see Usage on page 208.

Declarative Metadata Sample Definition

The following is an example of a CustomMetadata component. In this example, the sample app TravelApp deploys a Planets picklist, specifies its sort order, and adds picklist items to it.

Assuming Picklists R Us's namespace is `picklist1234`, to define the `Planets` picklist, TravelApp deploys a file in the `customMetadata` folder, named `picklist1234__ReusablePicklist.Planets.md`, with these contents. The `xsi:type` attribute specifies the type for the value of the `AlphaSort__c` checkbox field.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <description>All the planets in the solar system. Does not
                 include asteroids.</description>
    <label>Planets</label>
    <values>
        <field>picklist1234__AlphaSort__c</field>
        <value xsi:type="xsd:boolean">false</value>
    </values>
</CustomMetadata>
```

Picklists R Us creates its Reusable Picklist Option custom metadata type by deploying a file in the objects folder, named `ReusablePicklist__mdt.object`, with these contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fields>
        <fullName>Picklist__c</fullName>
        <externalId>false</externalId>
        <label>Picklist</label>
        <length>40</length>
        <required>true</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>SortOrder__c</fullName>
        <externalId>false</externalId>
        <label>Non-Alphabetical Sort Order</label>
        <precision>3</precision>
        <scale>0</scale>
        <required>false</required>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <label>Reusable Picklist Option</label>
    <pluralLabel>Reusable Picklist Options</pluralLabel>
</CustomObject>
```

To define the `Mars` picklist item, TravelApp deploys a file, named `picklist1234__ReusablePicklistOption.Mars.md`, with these contents. This component file specifies types that apply to the `ReusablePicklistOption__mdt` custom fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <label>Mars</label>
    <values>
        <field>picklist1234__Picklist__c</field>
```

```
        <value xsi:type="xsd:string">Planets</value>
    </values>
    <values>
        <field>picklist1234__SortOrder__c</field>
        <value xsi:type="xsd:int">4</value>
    </values>
</CustomMetadata>
```

To define the `Motel6` picklist item, TravelApp deploys a file, named `picklist1234__ReusablePicklistOption.Motel6.md`, with these contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <label>Motel 6</label>
    <values>
        <field>picklist1234__Picklist__c</field>
        <value xsi:type="xsd:string">Hotels</value>
    </values>
</CustomMetadata>
```

Because the `SortOrder__c` field isn't required, this file doesn't require a value for `SortOrder__c`. Alternatively, the file could have explicitly specified a value with `xsi:nil` to ensure that `SortOrder__c` was cleared of any previous value.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <label>Motel 6</label>
    <values>
        <field>picklist1234__Picklist__c</field>
        <value xsi:type="xsd:string">Hotels</value>
    </values>
    <values>
        <field>picklist1234__SortOrder__c</field>
        <value xsi:nil="true" />
    </values>
</CustomMetadata>
```

This excerpt from a `package.xml` file illustrates the inclusion of custom metadata types and their namespaces in custom metadata records' names. Assume that Picklists R Us's namespace is `picklist1234`.

```
<?xml version="1.0" encoding="UTF-8"?>
<package xmlns="http://soap.sforce.com/2006/04/metadata">
…
   <types>
      <members>picklist1234__ReusablePicklist.Hotels</members>
      <members>picklist1234__ReusablePicklist.Planets</members>
      <members>picklist1234__ReusablePicklistOption.Bellagio</members>
      <members>picklist1234__ReusablePicklistOption.Motel6</members>
      <members>picklist1234__ReusablePicklistOption.Mercury</members>
      <members>picklist1234__ReusablePicklistOption.Venus</members>
      <members>picklist1234__ReusablePicklistOption.Earth</members>
      <members>picklist1234__PicklistUsage.BookedHotel</members>
```

```
      <members>
         picklist1234__PicklistUsage.DestinationPlanetPL
      </members>
      <members>picklist1234__PicklistUsage.PlanetVisitedPl</members>
      <name>CustomMetadata</name>
   </types>
...
</package>
```

TravelApp, Inc.'s `package.xml` file uses a wildcard to install custom metadata, as is shown in this excerpt from their `package.xml` file. Unless you want to deploy or retrieve specific records, using a wildcard is easier than listing all of your custom metadata records in your `package.xml` file.

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

If the custom metadata is from a managed package, the name after the dot in the `package.xml` file—between the two dots in the file name—is qualified by the managed package's namespace. For example, assuming TravelApp uses the namespace `travelApp1234`, the first member element in the TravelApp `package.xml` file appears to Galactic Tours as:

```
<members>picklist1234__ReusablePicklist.travelApp1234__Hotels</members>
```

Usage

When specifying the `value` field in the CustomMetadataValue subtype, specify an appropriately typed object that's based on your field type definition. In declarative metadata definitions for CustomMetadataValue, use the `xsi:type` attribute of the value element. For example, to specify a boolean value: `<value xsi:type="xsd:boolean">true</value>`. Valid `xsi:type` attributes are:

Custom metadata value	Custom field definition
`xsi:type="xsd:boolean"`	Checkbox
`xsi:type="xsd:date"`	Date
`xsi:type="xsd:dateTime"`	Date/Time
`xsi:type="xsd:picklist"`	Picklist
`xsi:type="xsd:string"`	Text
`xsi:type="xsd:string"`	Phone
`xsi:type="xsd:string"`	TextArea
`xsi:type="xsd:string"`	URL
`xsi:type="xsd:string"`	Email
`xsi:type="xsd:int"`	Number/Percent, with scale equal to 0
`xsi:type="xsd:double"`	Number/Percent, with scale not equal to 0

You can also omit the `xsi:type` attribute. For example, `<value>true</value>`.

Although this attribute must be specified for any CustomMetadataValue, you can use an element with the `xsi:nil` attribute set to `true` to explicitly set the field's value to `null`. For example, `<value  xsi:nil="true"/>`.

Using `null` field values differs from leaving out the CustomMetadataValue for a particular field entirely. If you leave out the CustomMetadataValue, the value of the field doesn't change. The field's value is `null` for newly deployed custom metadata records and left at its previous value for updated custom metadata records.

When you retrieve CustomMetadataValue objects, the `value` field of the returned object holds a value of the correct type, specified by `xsi:type` in the case of declarative metadata definitions.

CustomObject

Represents a custom object that stores data unique to your organization or an external object that maps to data stored outside Salesforce. This type 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 Salesforce online help.

You can also use this metadata type to work with customizations of standard objects, such as accounts. For an example, see Standard Objects on page 16.

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

For example, the following are fully specified names for a standard object and a custom object respectively:

```
Account
MyCustomObject__c
```

And the following is a fully specified name for an external object:

```
MyExternalObject__x
```

For sample Java code that creates a custom object, see Step 3: Walk Through the Java Sample Code on page 6.

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.

External object names are automatically appended with __x. The file suffix is `.object` for the external object file.

Custom, standard, and external 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. External objects are available in API version 32.0 and later.

Fields

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

Field Name	Field Type	Description
`actionOverrides`	ActionOverride[]	A list of action overrides on the object. This field is available in API version 18.0 and later.
`allowInChatterGroups`	boolean	Indicates whether records of this custom object type can be added to Chatter groups. This field is availabe in API version 34.0 and later.
`businessProcesses`	BusinessProcess[]	A list of business processes associated with the object. This field is available in API version 17.0 and later.
`compactLayoutAssignment`	string	The compact layout assigned to the object. This field is available in API version 29.0 and later.
`compactLayouts`	CompactLayout[]	A list of compact layouts associated with the object. This field is available in API version 29.0 and later.
`customHelp`	string	The s-control that contains the help content if the object has customized help content. This field is available in API version 14.0 and later.
`customHelpPage`	string	The Visualforce page that contains the help content if the object has customized help content. This field is available in API version 16.0 and later.
`customSettingsType`	CustomSettingsType (enumeration of type string)	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: • `List`—static data stored in cache and accessed as part of your application and available organization-wide. • `Hierarchy`—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. This field is available in API version 17.0 and later.
`customSettingsVisibility`	CustomSettingsVisibility (enumeration of type string)	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: • `Public`—if the custom setting is packaged, it is accessible to all subscribing organizations. • `Protected`—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.

Field Name	Field Type	Description
		This field is available in API versions 17.0 through 33.0. In versions 34.0 and later, use the `visibility` field instead of this field.
`deploymentStatus`	DeploymentStatus (enumeration of type string)	Indicates the deployment status of the object.
`deprecated`	boolean	Reserved for future use.
`description`	string	A description of the object. Maximum of 1000 characters.
`enableActivities`	boolean	Indicates whether the object is enabled for activities (`true`) or not (`false`).
`enableBulkApi`	boolean	When enabled, the object is classified as an Enterprise Application object for usage tracking. When enabled, `enableSharing` and `enableStreamingApi` must also be enabled. This field is available in API version 31.0 and later.
`enableDivisions`	boolean	Indicates whether the object is enabled for divisions (`true`) or not (`false`). For more information about the Division object, see the *SOAP API Developer's Guide*.
`enableEnhancedLookup`	boolean	Indicates whether the object is enabled for enhanced lookups (`true`) or not (`false`). In API version 28.0 and later, this field can also be used for the Account, Contact, and User objects. 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 Salesforce online help.
`enableFeeds`	boolean	Indicates whether the object is enabled for feed tracking (`true`) or not (`false`). For more information, see "Customize Chatter Feed Tracking" in the Salesforce online help. This field is available in API version 18.0 and later.
`enableHistory`	boolean	Indicates whether the object is enabled for history tracking (`true`) or not (`false`). Also available for standard objects in API version 29.0 and later.
`enableReports`	boolean	Indicates whether the object is enabled for reports (`true`) or not (`false`).
`enableSearch`	boolean	Indicates whether the object can be searched via the data APIs (`true`) or not (`false`). Note: By default, search is disabled for new custom objects.

Field Name	Field Type	Description
		This field is available in API version 35.0 and later.
`enableSharing`	boolean	When enabled, the object is classified as an Enterprise Application object for usage tracking. When enabled, `enableBulkApi` and `enableStreamingApi` must also be enabled. This field is available in API version 31.0 and later.
`enableStreamingApi`	boolean	When enabled, the object is classified as an Enterprise Application object for usage tracking. When enabled, `enableBulkApi` and `enableSharing` must also be enabled. This field is available in API version 31.0 and later.
`externalDataSource`	string	Required and available for external objects only. The name of the external data source that stores the data for the external object. The data source is represented by the ExternalDataSource component. This field is available in API version 32.0 and later.
`externalName`	string	Required and available for external objects only. The name of the table in the external data source that contains the data for the external object. This field is available in API version 32.0 and later.
`externalRepository`	string	Available for Lightning Connect external objects only. Corresponds to `Display URL Reference Field` in the user interface. The external object's `Display URL` standard field values are automatically generated from the external system. For example, with the OData 2.0 adapter for Lightning Connect, the value is based on the `link href` that's defined on the OData producer. You can override the default values with the values of a custom field on the same external object. Select the field name, and make sure that the custom field's values are valid URLs. This field is available in API version 32.0 and later.
`externalSharingModel`	SharingModel(enumeration of type string)	Indicates the external organization-wide defaults for the object, which determines the access level for external users, such as portal and community users. This field is supported for these objects: • Accounts and their associated contracts and assets • Cases

Field Name	Field Type	Description
		• Contacts • Opportunities • Custom Objects • Users This field is available in API version 31.0 and later.
`fields`	CustomField[]	Represents one or more fields in the object.
`fieldSets`	FieldSet	Defines the field set that exists on this 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 `create()` to see an example of this field specified for a call. This value cannot be `null`.
`gender`	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender.
`household`	boolean	This field supports relationship groups, a feature available only with Salesforce for Wealth Management. For more information, see "Salesforce for Wealth Management Overview" in the Salesforce online help.
`historyRetentionPolicy`	HistoryRetentionPolicy	Reserved for future use.
`label`	string	Label that represents the object throughout the Salesforce user interface.
`listViews`	ListView[]	Represents one or more *list views* associated with the object.
`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 `lookupFilter` field in the CustomField component.
`nameField`	CustomField	Required for custom objects. External objects may instead specify the name field by setting `isNameField` to `true` in the CustomField component. 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 page layouts, related lists, lookup dialogs, search results, and

Field Name	Field Type	Description
		key lists on tab home pages. By default, this field is added to the custom object page layout as a required field.
`pluralLabel`	string	Plural version of the `label` value.
`recordTypes`	RecordType[]	An array of one or more record types defined for this object.
`recordTypeTrackFeedHistory`	boolean	Indicates whether the record type is enabled for feed tracking (`true`) or not (`false`). To set this field to `true`, the `enableFeeds` field on the associated CustomObject must also be `true`. For more information, see "Customize Chatter Feed Tracking" in the Salesforce online help. This field is available in API version 19.0 and later.
`recordTypeTrackHistory`	boolean	Indicates whether history tracking is enabled for this record type (`true`) or not (`false`). To set `recordTypeTrackHistory` to true the `enableHistory` field on the associated custom object must also be `true`. This field is available in API version 19.0 and later.
`searchLayouts`	SearchLayouts	The *Search Layouts* related list information for the object.
`sharingModel`	SharingModel(enumeration of type string)	Indicates the organization-wide defaults for the object. **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 Salesforce user interface. Using API version 30.0 and later, you can set this field for internal users using the API and the Salesforce user interface.
`sharingReasons`	SharingReason[]	The reasons why the object is being shared.
`sharingRecalculations`	SharingRecalculation[]	A list of custom sharing recalculations associated with the 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 the object.
`visibility`	SetupObjectVisibility (enumeration of type string)	When this field is present, this component is not a custom object, but a custom setting or custom metadata type. This field returns the visibility of the custom setting or custom metadata type. The following values are valid.

Field Name	Field Type	Description
		• `Public`—If the custom setting or custom metadata type is packaged, it's accessible to all subscribing organizations. • `Protected`—If the custom setting or custom metadata type is in a managed package, it's only accessible to the developer organization; subscribing organizations can't access it. The default value is `Public`. This field is available in API version 34.0 and later. For custom settings, this field replaces the `customSettingsVisibility` field.
`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:

- ActionOverride
- BusinessProcess
- CompactLayout
- CustomField
- FieldSet
- HistoryRetentionPolicy
- ListView
- RecordType
- SearchLayouts
- 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>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>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

The following is the metadata definition of an external object for Lightning Connect.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Delete</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Edit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Follow</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>List</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>New</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>SaveEdit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Tab</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <type>Default</type>
    </actionOverrides>
```

```
    <deploymentStatus>InDevelopment</deploymentStatus>
    <description>Products</description>
    <enableFeeds>false</enableFeeds>
    <externalDataSource>OData</externalDataSource>
    <externalIndexAvailable>false</externalIndexAvailable>
    <externalName>Products</externalName>
    <fields>
        <fullName>DiscontinuedDate__c</fullName>
        <description>DiscontinuedDate</description>
        <externalDeveloperName>DiscontinuedDate</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>DiscontinuedDate</label>
        <required>false</required>
        <type>DateTime</type>
    </fields>
    <fields>
        <fullName>ID__c</fullName>
        <description>ID</description>
        <externalDeveloperName>ID</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>ID</label>
        <precision>18</precision>
        <required>false</required>
        <scale>0</scale>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Name__c</fullName>
        <description>Name</description>
        <externalDeveloperName>Name</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Name</label>
        <length>128</length>
        <required>false</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Price__c</fullName>
        <description>Price</description>
        <externalDeveloperName>Price</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
```

```
        <isSortingDisabled>false</isSortingDisabled>
        <label>Price</label>
        <precision>16</precision>
        <required>false</required>
        <scale>2</scale>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Products__c</fullName>
        <externalDeveloperName>Products</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Products</label>
        <length>20</length>
        <referenceTo>Products__x</referenceTo>
        <relationshipLabel>Products</relationshipLabel>
        <relationshipName>Products</relationshipName>
        <type>ExternalLookup</type>
    </fields>
    <fields>
        <fullName>Rating__c</fullName>
        <description>Rating</description>
        <externalDeveloperName>Rating</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Rating</label>
        <precision>18</precision>
        <required>false</required>
        <scale>0</scale>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>ReleaseDate__c</fullName>
        <description>ReleaseDate</description>
        <externalDeveloperName>ReleaseDate</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>ReleaseDate</label>
        <required>false</required>
        <type>DateTime</type>
    </fields>
    <label>Products</label>
    <pluralLabel>Products</pluralLabel>
    <searchLayouts>
        <customTabListAdditionalFields>ExternalId</customTabListAdditionalFields>
        <lookupDialogsAdditionalFields>ExternalId</lookupDialogsAdditionalFields>
```

```
        <lookupPhoneDialogsAdditionalFields>ExternalId</lookupPhoneDialogsAdditionalFields>

        <searchResultsAdditionalFields>ExternalId</searchResultsAdditionalFields>
        <searchResultsAdditionalFields>DisplayUrl</searchResultsAdditionalFields>
        <searchResultsAdditionalFields>ID__c</searchResultsAdditionalFields>
    </searchLayouts>
</CustomObject>
```

SEE ALSO:

- CustomField
- Metadata
- Picklist (Including Dependent Picklist)
- SearchLayouts
- WebLink
- CustomObjectTranslation
- ListView
- CompactLayout

ActionOverride

Represents an action override on a standard or custom object. Use it to create, update, edit, or delete action overrides. You can only access ActionOverride by accessing its encompassing CustomObject.

Declarative Metadata File Suffix and Directory Location

Action overrides are defined as part of a standard or custom object.

Version

Action overrides are available in API version 18.0 and later. Beginning in Summer '13, action overrides can be applied to both standard or custom objects. Previously, action overrides only applied to custom objects.

Fields

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

Field Name	Field Type	Description
`actionName`	string	Required. The possible values are the same as the actions you can override: • `accept` • `clone` • `delete` • `edit` • `list` • `new`

Field Name	Field Type	Description
		• `tab` • `view`
`comment`	string	Any comments you want associated with the override.
`content`	string	Set this field if `type` is set to `scontrol` or `visualforce`. It refers to the name of the s-control or Visualforce page to use as the override. To reference installed components, use the format of ***`Component_namespace__Component_name`***.
`formFactor`	FormFactor (enumeration of type string)	If the `type` field is set to `flexipage`, set this field to `Large` to override the View action with a Lightning Page in Lightning Experience. The `Large` value represents the Lightning Experience desktop environment, and is only valid for the `flexipage` type. For the `scontrol` and `visualforce` types, this field defaults to `null`. This field is available in API version 37.0 and later, and is part of the pilot feature for creating and editing record pages in Lightning Experience. Note: Creating and editing Lightning Experience record pages using the Lightning App Builder is currently available to Developer Edition organizations through a pilot program. Pilot programs are subject to change, and as such, we cannot guarantee a particular time frame in which this feature can be enabled. Any unreleased services or features referenced in this document, press releases, or public statements are not currently available and may not be delivered on time or at all. Customers who purchase our services should make their purchase decisions based on features that are currently available.
`skipRecordTypeSelect`	boolean	Set this field to `true` if you prefer that any new records created by this action override aren't forwarded to the record type selection page. This field is only valid if the `actionName` is a "create" type (like `new`), and `type` is set to `visualforce`. This field is available in API version 21.0 and later.
`type`	ActionOverrideType (enumeration of type string)	Required. Represents the type of action override. Valid values are described in ActionOverrideType.

ActionOverrideType

ActionOverrideType is an enumeration of type string that defines which kind of action override to use. The valid values are:

- `default`—The override uses a custom override provided by an installed package. If there isn't one available, the standard Salesforce behavior is used.
- `flexipage`—The override uses behavior from a Lightning Page, and is only valid for use on the View standard button in Lightning Experience.
- `scontrol`—The override uses behavior from an s-control.

- `standard`—The override uses regular Salesforce behavior.
- `visualforce`—The override uses behavior from a Visualforce page.

Declarative Metadata Sample Definitions

You can define an action like this:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>visualforce</type>
        <content>myEditVFPage</content>
        <comment>This edit action is a lot safer.</comment>
     </actionOverrides>
</CustomObject>
```

With the previous definition, calling `retrieve()` presents:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>default</type>
     </actionOverrides>
</CustomObject>
```

If a subscriber installed a package with the previous metadata, you can override the behavior by editing the XML. For example, if you want the regular Salesforce behavior, use:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>standard</type>
     </actionOverrides>
</CustomObject>
```

To set a Lightning Page action override on the View standard button in Lightning Experience, use:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>View</actionName>
        <content>myLightningPage</content>
        <formFactor>Large</formFactor>
        <type>flexipage</type>
    </actionOverrides>
</CustomObject>
```

SEE ALSO:

CustomObject

BusinessProcess

The BusinessProcess metadata type enables you to display different picklist values for users based on their profile. This type extends the Metadata metadata type and inherits its `fullName` field.

Multiple business processes allow you to track separate sales, support, and lead lifecycles. A sales, support, lead, or solution process is assigned to a record type. The record type determines the user profiles that are associated with the business process. For more information, see " Managing Multiple Business Processes " in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Business processes are defined as part of the custom object or standard object definition. See CustomObject for more information.

Version

BusinessProcess components are available in API version 17.0 and later.

Fields

Field	Field Type	Description
`description`	string	Description for the business process.
`fullName`	string	The name used as a 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`isActive`	boolean	Indicates if the business process is active (`true`) or not (`false`).
`namespacePrefix`	string	The namespace of the developer organization where the package was created.
`values`	PicklistValue[]	A list of picklist values associated with this business process.

Declarative Metadata Sample Definition

A sample XML definition of a lead business process is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
    <businessProcesses>
        <fullName>HardwareLeadProcess</fullName>
        <description>Lead Process for hardware division</description>
        <isActive>true</isActive>
        <values>
            <fullName>Closed - Converted</fullName>
            <default>false</default>
        </values>
        <values>
            <fullName>CustomLeadStep1</fullName>
            <default>false</default>
```

```
        </values>
        <values>
            <fullName>CustomLeadStep2</fullName>
            <default>false</default>
        </values>
        <values>
            <fullName>Open - Not Contacted</fullName>
            <default>false</default>
        </values>
        <values>
            <fullName>Working - Contacted</fullName>
            <default>true</default>
        </values>
    </businessProcesses>
....
</CustomObject>
```

SEE ALSO:

CustomObject

CompactLayout

Represents the metadata associated with a compact layout. This type extends the Metadata metadata type and inherits its `fullName` field.

Compact layouts are used in Salesforce1 and Lightning Experience to display a record's key fields at a glance.

Compact layouts support all field types except:

- text area
- long text area
- rich text area
- multi-select picklist

For more information on compact layouts, see "Compact Layouts" in the Salesforce Help.

File Suffix and Directory Location

Compact layouts are defined as part of the custom object or standard object definition. See CustomObject for more information.

Version

CompactLayout components are available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`fields`	string	The fields assigned to the compact layout. Their order represents the prioritization given to them when defining the compact layout.

Field Name	Field Type	Description
label	string	Label that represents the object throughout the Salesforce user interface.

Declarative Metadata Sample Definition

The following is an example of a CompactLayout component:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>Accept</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Clone</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Delete</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Edit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>List</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>New</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Tab</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <type>Default</type>
    </actionOverrides>
    <compactLayouts>
        <fullName>testCompactLayout</fullName>
        <compactLayoutItems>textfield__c</compactLayoutItems>
        <label>testCompactLayoutLabel</label>
    </compactLayouts>
    <defaultCompactLayoutAssignment>SYSTEM</defaultCompactLayoutAssignment>
    <deploymentStatus>Deployed</deploymentStatus>
    <enableActivities>false</enableActivities>
    <enableFeeds>false</enableFeeds>
    <enableHistory>false</enableHistory>
    <enableReports>false</enableReports>
    <fields>
```

```
        <fullName>textfield__c</fullName>
        <externalId>false</externalId>
        <label>textfield</label>
        <length>255</length>
        <required>false</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <label>customObj</label>
    <nameField>
        <label>customObj Name</label>
        <type>Text</type>
    </nameField>
    <pluralLabel>customObjs</pluralLabel>
    <recordTypes>
        <fullName>RT1</fullName>
        <active>true</active>
        <label>RT1</label>
        <compactLayoutAssignment>testCompactLayout</compactLayoutAssignment>
    </recordTypes>
    <recordTypes>
        <fullName>RT2</fullName>
        <active>true</active>
        <label>RT2</label>
    </recordTypes>
    <searchLayouts/>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

CustomField

Represents the metadata associated with a field. Use this metadata type to create, update, or delete custom field definitions on standard, custom, and external objects or standard field definitions on standard objects. This type extends the Metadata metadata type and inherits its `fullName` field.

Only standard fields that you can customize are supported, that is, standard fields to which you can add help text or enable history tracking or Chatter feed tracking. Other standard fields aren't supported, including system fields (such as `CreatedById` or `LastModifiedDate`) and autonumber fields. All standard picklist fields are supported except for Lead.CampaignMemberStatus, Opportunity.ForecastCategoryName, and Order.Status.

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

```
MyCustomObject__c.MyCustomField__c
```

An example of a custom field on a standard object:

```
Account.MyAcctCustomField__c
```

An example of a standard field on a standard object:

```
Account.Phone
```

An example of a custom field on an external object:

```
MyExternalObject__x.MyCustomField__c
```

Note: In the Metadata API, external objects are represented by the CustomObject metadata type.

The following custom field types aren't available for external objects.

- Auto-number
- Currency
- Formula
- Geolocation
- Master-detail relationship
- Picklist
- Picklist (multi-select)
- Roll-up summary
- Text (encrypted)
- Text Area (rich)

Declarative Metadata File Suffix and Directory Location

Custom fields are user-defined fields and are part of the custom object or standard object definition. See CustomObject for more information. Standard fields are predefined on standard objects.

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 Fields on Custom or Standard Objects

When you retrieve a custom or standard object, you return everything associated with the object, except for standard fields that aren't customizable. You can also retrieve only specific 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`, each containing the requested field definitions.

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

Version

Custom and standard 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 (`true`) or not (`false`). For indirect lookup relationship fields on external objects, this attribute affects how this custom field's values are matched against the values of the `referenceTargetField`.
	string	
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 "Object Relationships" in the Salesforce 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 `true`, the geolocation appears in decimal notation. If `false`, the geolocation appears as degrees, minutes, and seconds.
encrypted	boolean	Note: This information applies to Platform Encryption and not to Classic Encryption. Indicates whether this field is encrypted (`true`) or not (`false`). This field is available in API version 34.0 and later.
externalDeveloperName	string	Available only for external objects. Name of the table column on the external data source that maps to this custom field in Salesforce. Corresponds to `External Column Name` in the user interface. This field is available in API version 32.0 and later.
externalId	boolean	Indicates whether the field is an external ID field (`true`) or not (`false`).
fieldManageability	string	Determines who can update the field after it's released in a managed package. Valid values: • Locked—The field can't be updated.

Field Name	Field Type	Description
		• DeveloperControlled—The creator of the record can update the field with a package upgrade. • SubscriberControlled—Anyone with proper permissions can update the field. The field can't be updated with a package upgrade. Available only for fields on custom metadata types.
`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 `BlankAsBlank` and `BlankAsZero`.
`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 `create()` to see an example of this field specified for a call. This value cannot be `null`.
`indexed`	boolean	Indicates if the field is indexed. If this field is unique or the `externalId` is set true, the `isIndexed` value is set to true. 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 "Define Field-Level Help" in the Salesforce Help.
`isFilteringDisabled`	boolean	Available only for external objects. Indicates whether the custom field is available in filters. This field is available in API version 32.0 and later.
`isNameField`	boolean	Available only for external object fields of type text. For each external object, you can specify one field as the name field. If you set this to `true`, make sure that the external table column identified by the `externalDeveloperName` attribute contains name values. This field is available in API version 32.0 and later.
`isSortingDisabled`	boolean	Available only for external objects. Indicates whether the custom field is sortable. This field is available in API version 32.0 and later.
`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 `false`. 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 `Industry` field for accounts.
`length`	int	Length of the field.

Field Name	Field Type	Description
`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. Note: LookupFilter is not supported on the article type object.
`maskChar`	EncryptedFieldMaskChar (enumeration of type string)	Note: This information applies to Classic Encryption and not to Platform Encryption. 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 "Classic Encryption for Custom Fields" in the Salesforce Help.
`maskType`	EncryptedFieldMaskType (enumeration of type string)	Note: This information applies to Classic Encryption and not to Platform Encryption. 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 "Classic Encryption for Custom Fields" in the Salesforce 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 (`true`) or not (`false`).
`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.
`referenceTargetField`	string	Available only for indirect lookup relationship fields on external objects. Specifies the custom field on the parent object to match against this indirect lookup relationship field, whose values come from an external data source. The specified custom field on the parent object must have both `externalId` and `unique` set to `true`. This field is available in API version 32.0 and later.
`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 MyObject that had a relationship to YourObject, the relationship name might be YourObjects.
`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

Field Name	Field Type	Description
		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 and feel, and record ownership for junction objects. For more information, see the Salesforce 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 (`true`) or not (`false`).
`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. When you create records, `Starting Number`'s value increments to store the number that will be assigned to the next auto-number field created. **Note:** • You can't retrieve the starting number of an auto-number field through Metadata API. To specify a `Starting Number` while deploying, add a `startingNumber` tag for your field to your `package.xml` file. For example: `<startingNumber>42</startingNumber>` • If you deploy without specifying a `Starting Number` value in your `package.xml` file, the default starting number for standard fields is `0`. The default starting number for custom fields is `1`.
`stripMarkup`	boolean	Set to `true` to remove markup, or `false` 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 `count`.
`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.

Field Name	Field Type	Description
`trackFeedHistory`	boolean	Indicates whether the field is enabled for feed tracking (`true`) or not (`false`). To set this field to `true`, the `enableFeeds` field on the associated CustomObject must also be `true`. For more information, see "Customize Chatter Feed Tracking" in the Salesforce Help. This field is available in API version 18.0 and later.
`trackHistory`	boolean	Indicates whether history tracking is enabled for the field (`true`) or not (`false`). Also available for standard object fields (picklist and lookup fields only) in API version 30.0 and later. To set `trackHistory` to `true`, the `enableHistory` field on the associated standard or custom object must also be `true`. For more information, see "Track Field History" in the Salesforce Help. Field history tracking isn't available for external objects.
`trackTrending`	boolean	Indicates whether historical trending data is captured for the field (`true`) or not (`false`). An object is enabled for historical trending if this attribute is `true` for at least one field. Available in API version 29.0 and later. For more information, see "Report on Historical Changes" in the Salesforce 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	Indicates the field type for the field. Valid values are enumerated in FieldType. For standard fields on standard objects, the `type` field is optional. This field is included for some standard field types, such as Picklist or Lookup, but not for others. The `type` field is included for custom fields.
`unique`	boolean	Indicates whether the field is unique (`true`) or not (`false`).
`visibleLines`	int	Indicates the number of lines displayed for the field.
`writeRequiresMasterRead`	boolean	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. • `true`—Allows users with "Read" access to the master record permission to create, edit, or delete child records. This setting makes sharing less restrictive.

Field Name	Field Type	Description
		• `false`—Allows users with "Read/Write" access to the master record permission to create, edit, or delete child records. This setting is more restrictive than `true`, and is the default value. For junction objects, the most restrictive access from the two parents is enforced. For example, if you set to `true` 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.

Fields use additional data types. For more information, see Metadata Field Types on page 261.

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 Classic Encryption for Custom Fields in the Salesforce 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 Salesforce.

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

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

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

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

`nino`
All characters are hidden. Salesforce 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 Salesforce.

For more information on encrypted fields, see Classic Encryption for Custom Fields in the Salesforce 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
`active`	boolean	Required. Indicates whether or not the lookup filter is active.
`booleanFilter`	string	Specifies advanced filter conditions. For more information on advanced filter conditions, see "Getting the Most Out of Filter Logic" in the Salesforce Help.
`description`	string	A description of what this filter does.
`errorMessage`	string	The error message that appears if the lookup filter fails.
`filterItems`	FilterItem[]	Required. The set of filter conditions. You can have up to 10 FilterItems per lookup filter.
`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.
`isOptional`	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
`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 `my_number_field__c > 1`, the value of `value` is `1`.
`valueField`	string	Specifies if the final column in the filter contains a field or a field value. Approval processes don't support `valueField` entries in filter criteria.

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
- within (DISTANCE criteria only)

SummaryOperations

Represents the type of a summaryOperation. Valid values are:

- Count
- Min
- Max
- Sum

Declarative Metadata Sample Definition

The following example shows a field definition for a custom field that is named Comments__c.

```
<?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 help text for this object</inlineHelpText>
        <label>Comments</label>
        <length>32000</length>
        <type>LongTextArea</type>
        <visibleLines>30</visibleLines>
</fields>
....
</CustomObject>
```

The following is the definition for two fields on the Account standard object—a custom field (MyCustomAccountField__c), and a standard field (Phone) that has history tracking enabled.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fields>
        <fullName>MyCustomAccountField__c</fullName>
        <description>A custom field on the Account standard object.</description>
        <externalId>false</externalId>
        <inlineHelpText>Some help text.</inlineHelpText>
        <label>MyCustomAccountField</label>
        <length>100</length>
        <required>false</required>
        <trackFeedHistory>false</trackFeedHistory>
```

```
        <trackHistory>false</trackHistory>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Phone</fullName>
        <trackFeedHistory>false</trackFeedHistory>
        <trackHistory>true</trackHistory>
    </fields>
</CustomObject>
```

SEE ALSO:

- CustomObject
- Picklist (Including Dependent Picklist)
- Metadata
- NamedFilter

FieldSet

Represents a field set. A field set is a grouping of fields. For example, you could have a field set that contains fields describing a user's first name, middle name, last name, and business title. Field sets can be referenced on Visualforce pages dynamically. If the page is added to a managed package, administrators can add, remove, or reorder fields in a field set to modify the fields presented on the Visualforce page without modifying any code.

Version

FieldSet components are available in API version 21.0 and later.

Fields

Field	Field Type	Description
availableFields	FieldSetItem[]	An array containing all the possible fields in the field set.
description	string	Required. A description provided by the developer that describes the field set. This is required.
displayedFields	FieldSetItem[]	An array containing all the fields that are presented on the Visualforce page. The order in which a field is listed determines the order of appearance on the page.
label	string	Required. The label used to reference the field set.

FieldSetItem

FieldSetItem represents an individual field in a field set.

Field	Field Type	Description
field	string	Required. The name of a field in a standard or custom object.
isFieldManaged	boolean	Read-only. Denotes whether the field was added to the field set via a managed or unmanaged package.
isRequired	boolean	Read-only. Indicates whether the field is universally required (true) or not (false).

Declarative Metadata Sample Definition

A sample XML definition of a FieldSet component is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fieldSets>
        <fullName>FieldSetNames</fullName>
        <availableFields>
            <field>MiddleName__c</field>
        </availableFields>
        <availableFields>
            <field>Title__c</field>
        </availableFields>
        <description>FieldSet containing how to properly address someone</description>
        <displayedFields>
            <field>FirstName__c</field>
        </displayedFields>
        <displayedFields>
            <field>LastName__c</field>
        </displayedFields>
        <label>FieldSet Names</label>
    </fieldSets>
</CustomObject>
```

HistoryRetentionPolicy

Represents the policy for retaining field history data. By setting a policy, you can specify the number of months you want to maintain field history in Salesforce, and the number of years that you want to retain field history in the archive.

This component is only available to users with the “RetainFieldHistory” permission.

Declarative Metadata File Suffix and Directory Location

Field history retention policies are defined as part of a standard or custom object. You can set field history retention policies for objects individually. See CustomObject for more information.

Version

Available in API version 31.0 and later.

Fields

Field Name	Field Type	Description
`archiveAfterMonths`	int	Required. The number of months that you want to keep field history data in Salesforce before archiving. You can set a minimum of 1 month and a maximum of 18 months. If you don't set a number, the default is 18 months. (That is, Salesforce maintains data for 18 months before archiving.)
`archiveRetentionYears`	int	Required. The number of years that you want to retain data in the archive. You can set a minimum of zero years, and a maximum of 10 years. If no number is set, the default is 10 years.
`description`	string	A text description for the history retention.
`gracePeriodDays`	int	The number of days of extra time after the `archiveAfterMonths` period before the data is archived. The `gracePeriodDays` interval applies only to the first time that the data is archived; because all the data is copied the first time, the operation may take longer than subsequent times when only the data that changed since the last archival operation is copied. The `gracePeriodDays` provides extra time for the administrator to prepare the organization before the initial archive operation. You can set a minimum of zero days and a maximum of 10 days. If no number is set, the default is 1 day.

Declarative Metadata Sample Definition

This sample shows the definition of a history retention policy for a custom object:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
   <historyRetentionPolicy>
       <archiveAfterMonths>6</archiveAfterMonths>
       <archiveRetentionYears>5</archiveRetentionYears>
       <description>My field history retention</description>
   </historyRetentionPolicy>
   <fields>
       <fullName>AccountSource</fullName>
...
</CustomObject>
```

ListView

ListView allows you to see a filtered list of records, such as contacts, accounts, or custom objects. This type extends the Metadata metadata type and inherits its `fullName` field. See "Create Custom List Views in Salesforce Classic" in the Salesforce online help.

Note: List views with the Visible only to me `Restrict Visibility` option are not accessible in Metadata API. Each of these list views is associated with a particular user.

Declarative Metadata File Suffix and Directory Location

List views are stored within a CustomObject component. The component can represent a custom object or a standard object, such as an account.

Version

ListView components for custom objects are available in API version 14.0 and later. ListView components for standard objects, such as accounts, are available in API version 17.0 and later.

Fields

Field	Field Type	Description
`booleanFilter`	string	This field represents an Advanced Option for a filter. Advanced Options in filters allow you to build up filtering conditions that use a mixture of AND and OR boolean operators across multiple filter line items. For example, `(1 AND 2) OR 3` finds records that match both the first two filter line items or the third. See "Getting the Most Out of Filter Logic" in the Salesforce online help.
`columns`	string[]	The list of fields in the list view. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field. Note: Field names in the ListView columns don't always match their API name counterparts. In particular, if person accounts is enabled in your organization, standard fields merged from a contact into an account start with the `PC_` prefix, while the corresponding API name starts with the `Person` prefix. For example, the ListView column name is `PC_Email` for a corresponding API field name of `PersonEmail`.
`division`	string	If your organization uses divisions to segment data and you have the "Affected by Divisions" permission, records in the list view must match this division. This field is only available if you are searching all records. This field is available in API version 17.0 and later.
`filterScope`	FilterScope (enumeration of type string)	Required. This field indicates whether you are filtering by owner or viewing all records.
`filters`	ListViewFilter[]	The list of filter line items.
`fullName`	string	Required. 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.

Field	Field Type	Description
`label`	string	Required. The list view name.
`language`	Language	The language used for filtering if your organization uses the Translation Workbench and you are using the `startsWith` or `contains` operator. The values entered as search terms must be in the same language as the filter language. See "Enter Filter Criteria" in the Salesforce online help. For a list of valid language values, see Language. This field is available in API version 17.0 and later.
`queue`	string	The name of a queue. Objects are sometimes assigned to a queue so that the users who have access to the queue can monitor and manage them. When you create a queue, a corresponding list view is automatically created. See "Create Queues" in the Salesforce online help.
`sharedTo`	SharedTo	Sharing access for the list view. This field is available in API version 17.0 and later.

ListViewFilter

ListViewFilter represents a filter line item.

Field	Field Type	Description
`filter`	string	Required. Represents the field specified in the filter.
`operation`	FilterOperation (enumeration of type string)	Required. The operation used by the filter, such as `equals`. The valid values are listed in FilterOperation.
`value`	string	Represents the value of the filter item being operated upon, for example, if the filter is `my_number_field__c > 1`, the value of `value` is `1`.

FilterScope

This is an enumeration of type string that represents the filtering criteria for the records. The valid values are listed in the table below:

Enumeration Value	Description
`Everything`	All records, for example All Opportunities.
`Mine`	Records owned by the user running the list view, for example My Opportunities.
`Queue`	Records assigned to a queue.
`Delegated`	Records delegated to another user for action: for example, a delegated task. This option is available in API version 17.0 and later.

Enumeration Value	Description
MyTerritory	Records in the territory of the user seeing the list view. This option is available if territory management is enabled for your organization. This option is available in API version 17.0 and later.
MyTeamTerritory	Records in the territory of the team of the user seeing the list view. This option is available if territory management is enabled for your organization. This option is available in API version 17.0 and later.
Team	Records assigned to a team. This option is available in API version 17.0 and later.

Declarative Metadata Sample Definition

A sample XML definition of a list view in a custom object is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
. . .
    <listViews>
        <fullName>All_Mileages</fullName>
        <filterScope>all</filterScope>
        <label>All Mileages</label>
    </listViews>
    <listViews>
        <fullName>My_Mileages</fullName>
        <booleanFilter>1 AND 2</booleanFilter>
        <columns>NAME</columns>
        <columns>CREATED_DATE</columns>
        <filterScope>mine</filterScope>
        <filters>
            <field>NAME</field>
            <operation>equals</operation>
            <value>Eric Bristow</value>
        </filters>
        <filters>
            <field>City__c</field>
            <operation>equals</operation>
            <value>Paris</value>
        </filters>
        <label>My Mileages</label>
    </listViews>
. . .
</CustomObject>
```

SEE ALSO:

CustomObject

Sample package.xml Manifest Files

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. This type 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
`active`	boolean	Required. Indicates whether or not the lookup filter is active.
`booleanFilter`	string	Specifies advanced filter conditions. For more information on advanced filter conditions, see "Getting the Most Out of Filter Logic" in the Salesforce online help.
`description`	string	A description of what this filter does.
`errorMessage`	string	The error message that appears if the lookup filter fails.
`field`	string	Required. The `fullName` 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,

Field Name	Field Type	Description
		updating, or deleting. See `create()` to see an example of this field specified for a call. This value cannot be `null`.
`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 `name` 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 `my_number_field__c > 1`, the value of `value` 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, such as an account.

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, such as accounts, are available in API version 16.0 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 `fullName` 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 (`true`), or not (`false`).

PicklistValue

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

- When you retrieve a standard object, all picklist values are retrieved, not just the customized picklist values.
- When you deploy changes to standard picklist fields, picklist values are added 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.
- Starting with API version 27.0, if picklist values are missing from a component definition, they get deleted from Salesforce when deployed. The deletion occurs for picklist values of standard and custom fields.

Field Name	Field Type	Description
`allowEmail`	boolean	Indicates whether this value lets users email a quote PDF (`true`), or not (`false`). This field is only relevant for the `Status` 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 (`true`), or not (`false`). This field is only relevant for the standard `Status` 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 #FF6600. 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: • `Checkbox`: `checked` or `unchecked`. • `Picklist`: The `fullName` of the picklist value in the controlling field.

Field Name	Field Type	Description
converted	boolean	Indicates whether this value is associated with a converted status (true), or not (false). This field is relevant for only the standard Lead Status 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 that you want to forecast. For more information, see "Convert Qualified Leads" in the Salesforce online help. This field is available in API version 16.0 and later.
cssExposed	boolean	Indicates whether this value is available in your Self-Service Portal (true), or not (false). This field is only relevant for the standard Case Reason field in cases. 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 Salesforce online help. 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. This field is available in API version 16.0 and later.
default	boolean	Required. Indicates whether this value is the default picklist value in the specified picklist (true), or not (false).
description	string	Description of a custom picklist value. This field is only relevant for the standard Stage 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)	Indicates whether this value is associated with a forecast category (true), or not (false). This field is only relevant for the standard Stage field in opportunities. For more information about forecast categories, including the valid string values listed below, see " Working with Forecast Categories " in the Salesforce online help. • Omitted • Pipeline • BestCase • Forecast • Closed This field is available in API version 16.0 and later.
fullName	string	The name used as a 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

Field Name	Field Type	Description
highPriority	boolean	Indicates whether this value is a high priority item (true), or not (false). This field is only relevant for the standard Priority field in tasks. For more information about tasks, see "Considerations for Using Tasks" in the Salesforce online help. This field is available in API version 16.0 and later.
probability	int	Indicates whether this value is a probability percentage (true), or not (false). This field is only relevant for the standard Stage field in opportunities. 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 Salesforce 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 Salesforce 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 (true), or not (false). This field is only relevant for the standard Status field in solutions. For more information about opportunities, see "Creating Solutions" in the Salesforce 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 (true), or not (false). This field is only relevant for the standard Stage 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 446.

```
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>
        <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>
```

RecordType

Represents the metadata associated with a record type. Record types let you offer different business processes, picklist values, and page layouts to different users. For more information, see "Record Types" in the Salesforce online help. Use this metadata type to create, update, or delete record type definitions for a custom object. This type extends the Metadata metadata type and inherits its `fullName` field.

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

Record types are available in API version 12.0 and later.

Fields

Field	Field Type	Description
`active`	boolean	Required. Indicates whether or not the record type is active.

Field	Field Type	Description
`businessProcess`	string	The `fullName` of the business process associated with the record type. This field is required in record types for lead, opportunity, solution, and case, and not allowed otherwise. See BusinessProcess on page 221. This field is available in API version 17.0 and later.
`compactLayoutAssignment`	string	Represents the compact layout that is assigned to the record type. This field is available in API version 29.0 and later.
`description`	string	Record type description. Maximum of 255 characters.
`fullName`	string	Record type name. The `fullName` 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. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the `label` field. 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. This value cannot be `null`.
`label`	string	Required. Descriptive label for the record type. The list of characters allowed in the `fullName` field has been reduced for versions 14.0 and later. This field contains the value contained in the `fullName` field before version 14.0.
`picklistValues`	RecordTypePicklistValue[]	Represents a set of values for a picklist.

RecordTypePicklistValue

RecordTypePicklistValue represents the combination of picklists and valid values that define a record type:

Field Name	Field Type	Description
`picklist`	string	Required. The name of the picklist.
`values`	PicklistValue	One or more of the picklist values in the picklist. Each value defined is available in the record type that contains this component.

Java Sample

The following sample uses two record types. For the complete sample that includes profiles and picklists, see Profile on page 446.

```
public void recordTypeSample() {
  try {
    // Employees and managers have different access
    // to the state of the expense sheet
    RecordType edit = new RecordType();
    edit.setFullName("ExpenseReport__c.Edit");
    edit.setLabel("ExpenseReport__c.Label");
    PicklistValue unsubmitted = new PicklistValue();
    unsubmitted.setFullName("Unsubmitted");
    PicklistValue submitted = new PicklistValue();
    submitted.setFullName("Submitted");
    RecordTypePicklistValue editStatuses =
        new RecordTypePicklistValue();
    editStatuses.setPicklist("ExpenseStatus__c");
    editStatuses.setValues(
        new PicklistValue[] {unsubmitted, submitted});
    edit.setPicklistValues(
        new RecordTypePicklistValue[] {editStatuses});
    AsyncResult[] arsEdit =
        metadataConnection.create(new Metadata[] {edit});

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

Declarative Metadata Sample Definition

The definition of a record type in a custom object is shown below:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
. . .
   <recordTypes>
        <fullName>My First Recordtype</fullName>
```

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

SearchLayouts

Represents the metadata associated with the Search Layouts for an object. You can customize which fields to display for users in search results, search filter fields, lookup dialogs, and recent record lists on tab home pages. For more information, see "Customize Search Layouts" and "Customize Search Layouts for Custom Objects" in the Salesforce online help.

Version

Search layouts for custom objects are available in API version 14.0 and later. The ability to modify search layouts for standard objects (except events and tasks) is available in API version 27.0 and later.

Fields

Field	Field Type	Description
`customTabListAdditionalFields`	string[]	The list of fields displayed in the Recent *`Object Name`* list view on a tab associated with the object. The `name` field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field.
`excludedStandardButtons`	string[]	The list of standard buttons excluded from the search layout.
`listViewButtons`	string[]	The list of buttons available in list views for an object. This field is equivalent to the Buttons Displayed value in the *`Object Name`* `List View` in the Search Layouts related list on the object detail page in the Salesforce user interface. For more information, see "Lookup Dialog Search" in the Salesforce online help.
`lookupDialogsAdditionalFields`	string[]	The list of fields displayed in a lookup dialog for the object. The `name` field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field. Salesforce objects often include one or more *lookup fields* that allow users to associate two records together in a relationship. For example, a contact record includes an `Account` lookup field that represents the relationship between the contact and the organization with which the contact is associated. A lookup search dialog helps you search

Field	Field Type	Description
		for the record associated with the one being edited. Lookup filter fields allow you to filter your lookup search by a customized list of fields in the object. This field is equivalent to the `Lookup Dialogs` in the Search Layouts related list on the object detail page in the application user interface. For more information, see "Lookup Dialog Search" in the Salesforce online help.
`lookupFilterFields`	string[]	The list of fields that can be used to filter enhanced lookups for an object. Enhanced lookups are optionally enabled by your administrator. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field. This field is equivalent to the `Lookup Filter Fields` in the Search Layouts related list on the object detail page in the application user interface. For more information, see "Lookup Dialog Search" in the Salesforce online help.
`lookupPhoneDialogsAdditionalFields`	string[]	The list of phone-related fields displayed in a lookup dialog for the object. The `name` field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field. This list enables integration of the fields with a SoftPhone dial pad. For more information, see "About CTI 1.0 and 2.0 SoftPhones" in the Salesforce online help. This field is equivalent to the `Lookup Phone Dialogs` in the Search Layouts related list on the object detail page in the application user interface.
`searchFilterFields`	string[]	The list of fields that can be used to filter a search for the object. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field. This field is equivalent to the `Search Filter Fields` in the Search Layouts related list on the object detail page in the application user interface.
`searchResultsAdditionalFields`	string[]	The list of fields displayed in a search result for the object. The `name` field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field.

Field	Field Type	Description
		This field is equivalent to the `Search Results` in the Search Layouts related list on the object detail page in the application user interface.
`searchResultsCustomButtons`	string[]	The list of custom buttons available in a search result for the object. The actions associated with the buttons can be applied to any of the records returned in the search result.

Declarative Metadata Sample Definition

A sample definition of search layouts in an object is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
. . .
    <searchLayouts>
        <listViewButtons>New</listViewButtons>
        <listViewButtons>Accept</listViewButtons>
        <listViewButtons>ChangeOwner</listViewButtons>
        <lookupDialogsAdditionalFields>firstQuote__c</lookupDialogsAdditionalFields>
        <lookupDialogsAdditionalFields>finalQuote__c</lookupDialogsAdditionalFields>
        <searchResultsAdditionalFields>CREATEDBY_USER</searchResultsAdditionalFields>
     </searchLayouts>
. . .
</CustomObject>
```

SEE ALSO:

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 “Sharing Settings” in the Salesforce online help.

Use SharingReason to create, update, or delete sharing reason definitions for a custom object. This type 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
`fullName`	string	Required. Sharing reason name. The __c suffix is appended to custom sharing reasons. 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.
`label`	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 "Recalculate Apex Managed Sharing" in the Salesforce online help.

Version

Sharing recalculations are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`className`	string	Required. The Apex class that recalculates the Apex sharing for a custom object. This class must implement the `Database.Batchable` 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. This type 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
`active`	boolean	Required. Indicates whether this validation rule is active, (`true`), or not active (`false`).
`description`	string	A description of the validation rule.
`errorConditionFormula`	string	Required. The formula defined in the validation rule. If the formula returns a value of `true`, an error message is displayed. See "Define Validation Rules" in the Salesforce online help.
`errorDisplayField`	string	The fully specified name of a field in the application. If a value is supplied, the error message appears next to the specified field. If you do not specify a value or the field isn't visible on the page layout, the value changes automatically to `Top of Page`.
`errorMessage`	string	Required. The message that appears if the validation rule fails. The message must be 255 characters or less.
`fullName`	string	The internal name of the object. White spaces and special characters are escaped for validity. The name must: • Contain characters, letters, or the underscore (_) character

Field Name	Field Type	Description
		• Must start with a letter • Can't end with an underscore • Can't 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.

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. This type 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: • `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 it if your template requires data in a different format. This is available if your Content Source is URL. Valid values include: • `UTF-8`—Unicode (UTF-8) • `ISO-8859-1`—General US & Western Europe (ISO-8859–1, ISO-LATIN-1) • `Shift_JIS`—Japanese (Shift-JIS) • `ISO-2022-JP`—Japanese (JIS) • `EUC-JP`—Japanese (EUC-JP) • `x-SJIS_0213`—Japanese (Shift-JIS_2004) • `ks_c_5601-1987`—Korean (ks_c_5601-1987) • `Big5`—Traditional Chinese (Big5) • `GB2312`—Simplified Chinese (GB2312) • `Big5-HKSCS`—Traditional Chinese Hong Kong (Big5–HKSCS)
`fullName`	string	The name of the WebLink with white spaces and special characters escaped for validity. 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.
`hasMenubar`	boolean	If the `openType` is `newWindow`, whether to show the browser menu bar for the window (`true` or not (`false`). Otherwise this field should not be specified.

Field Name	Field Type	Description
`hasScrollbars`	boolean	If the `openType` is `newWindow`, whether to show the scroll bars for the window (`true`) or not (`false`). Otherwise this field should not be specified.
`hasToolbar`	boolean	If the `openType` is `newWindow`, whether to show the browser toolbar for the window (`true`) or not (`false`). Otherwise this field should not be specified.
`height`	int	Height in pixels of the window opened by this WebLink. Required if the `openType` is `newWindow`. Otherwise, this field should not be specified.
`isResizable`	boolean	If the `openType` is `newWindow`, whether to allow resizing of the window (`true`) or not (`false`). Otherwise this field should not be specified.
`linkType`	WebLinkType (enumeration of type string)	Required. Represents whether the content of this WebLink is specified by a URL, an sControl, a JavaScript code block, or a Visualforce page. • `url` • `sControl` • `javascript` • `page` • `flow`—Reserved for future use.
`masterLabel`	string	Master label for this object. This display value is the internal label that is not translated.
`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: • `newWindow` • `sidebar` • `noSidebar` • `replace` • `onClickJavaScript`
`page`	string	If the value of `linkType` is `page`, this field represents the Visualforce page; otherwise, this field should not be specified.
`position`	WebLinkPosition (enumeration of type string)	If the `openType` is `newWindow`, how the new window should be displayed. Otherwise this field should not be specified. Valid values: • `fullScreen` • `none` • `topLeft`
`protected`	boolean	Required. Indicates whether this sub-component is protected (`true`) or not (`false`). Protected sub-components cannot be linked to or

Field Name	Field Type	Description
		referenced by components or sub-components created in the installing organization.
requireRowSelection	boolean	If the openType is massAction, whether to require individual row selection to execute the action for this button (true) or not (false). Otherwise this field should not be specified.
scontrol	string	If the value of linkType is sControl, this field represents the name of the sControl; otherwise, this field should not be specified.
showsLocation	boolean	If the openType is newWindow, whether to show the browser location bar for the window (true) or not (false); otherwise this field should not be specified.
showsStatus	boolean	If the openType is newWindow, 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 linkType is url, this is the URL value. If the value of linkType is javascript, 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 openType is newWindow, 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. For related samples, see HomePageComponent and HomePageLayout.

```
<?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>
```

SEE ALSO:

- HomePageComponent
- HomePageLayout
- CustomPageWebLink

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.
DeleteConstraint	Custom field	A string that represents deletion options for lookup relationships. Valid values are: • `SetNull` • `Restrict` • `Cascade`
DeploymentStatus	Custom object Custom field	A string which represents the deployment status of a custom object or field. Valid values are: • `InDevelopment` • `Deployed`
FieldType	Custom field	Indicates the type of a custom field. Valid values are: • `AutoNumber` • `Lookup` • `MasterDetail` • `Checkbox` • `Currency` • `Date` • `DateTime` • `Email` • `EncryptedText` Note: This information applies to Classic Encryption and not to Platform Encryption. • `ExternalLookup` • `IndirectLookup` • `Number`[1] • `Percent` • `Phone` • `Picklist` • `MultiselectPicklist` • `Summary` • `Text` • `TextArea` • `LongTextArea` • `Summary` • `Url` • `Hierarchy` • `File`

Field Type	Objects	What the Field Contains
		• `CustomDataType` • `Html` • `Geolocation` [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.
Gender	Custom object	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender. Valid values are: • `Masculine` • `Feminine` • `Neuter` • `AnimateMasculine` (Slavic languages—currently Czech, Polish, Russian, Slovak, Slovenian, and Ukrainian) Note: The following genders are displayed on the Rename Tabs and Labels page in Setup but are stored internally as "Feminine". When setting them through the Metadata API, use "Feminine". • `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	Represents the sharing model for the custom object. Depending on the object, valid values are: • `Private` • `Read` • `ReadWrite` • `ReadWriteTransfer` • `FullAccess` • `ControlledByParent` For example, the User object supports `Private` and `Read` values. Accounts, opportunities, and custom objects support `Private`, `Read` and `ReadWrite` values.
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: • `Consonant` • `Vowel` • `Special (for nouns starting with z, or s plus consonants)`

Field Type	Objects	What the Field Contains
TreatBlanksAs	Custom field	Indicates how blanks should be treated. Valid values are: • `BlankAsBlank` • `BlankAsZero`

CustomObjectTranslation

This metadata type allows you to translate custom objects for a variety of languages. This type extends the Metadata metadata type and inherits its `fullName` field. The ability to translate component labels is part of the Translation Workbench. For more information, see "Enable and Disable the Translation Workbench" in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Translations are stored in a file with a format of *`customObjectName__c-lang`*`.objectTranslation`, where *`customObjectName__c`* is the custom object name, and *`lang`* is the translation language. A sample file name for German translations is `myCustomObject__c-de.objectTranslation`.

Custom object translations are stored in the `objectTranslations` folder in the corresponding package directory.

Version

CustomObjectTranslation components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`caseValues`	ObjectNameCaseValue[]	Different combinations of the custom object with regard to article, plural, possessive, and case.
`fields`	CustomFieldTranslation[]	A list of translations for the custom fields associated with the custom object.
`fullName`	string	The name of the custom object and the translation language with a format of *`customObjectName-lang`*, where *`customObjectName`* is the custom object name, and *`lang`* is the translation language. 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.
`gender`	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender.

Field	Field Type	Description
layouts	LayoutTranslation[]	A list of page layout translations.
nameFieldLabel	string	The label for the name field. Maximum of 765 characters.
namedFilters	NamedFilterTranslation[]	A list of translations for lookup filter error messages associated with the custom object. This field has been removed as of API version 30.0 and is only available in prior versions. The translation metadata associated with a lookup filter is now represented by the `lookupFilter` field in the CustomFieldTranslation subtype.
quickActions	QuickActionTranslation[]	A list of translations for actions.
recordTypes	RecordTypeTranslation[]	A list of record type translations.
sharingReasons	SharingReasonTranslation[]	A list of sharing reason translations.
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.
validationRules	ValidationRuleTranslation[]	A list of validation rule translations.
webLinks	WebLinkTranslation[]	A list of web link translations.
workflowTasks	WorkflowTaskTranslation[]	A list of workflow task translations.

CustomFieldTranslation

CustomFieldTranslation contains details for a custom field translation. For more details, see CustomField.

Note: Not every language supports all the possible values for the fields in CustomFieldTranslation. For language-specific supported values, see the fully supported languages and end-user languages appendices.

Field	Field Type	Description
caseValues	ObjectNameCaseValue[]	Different combinations of the custom object with regard to article, plural, possessive, and case. Available in API version 29.0 and later.
description	string	Translation for the custom field description.
gender	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender. Available in API version 29.0 and later.
help	string	Translation for the text that displays in the field-level help hover text for this field.
label	string	Translation for the label. Maximum of 765 characters.

Field	Field Type	Description
`lookupFilter`	LookupFilterTranslation	Represents the translation metadata associated with a lookup filter. This field is available in API version 30.0 and later. Note: LookupFilter is not supported on the article type object.
`name`	string	Required. The name of the field relative to the custom object; for example, `MyField__c`.
`picklistValues`	PicklistValueTranslation[]	List of translations for picklist values. See PicklistValue.
`relationshipLabel`	string	Translation for a lookup relationship label. A lookup relationship allows a field to be associated with another field. The relationship field allows users to select an option from a list of values defined by the other field. Maximum of 765 characters.
`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. Available in API version 29.0 and later.

LayoutTranslation

LayoutTranslation contains details for a page layout translation. For more details, see Fields.

Field	Field Type	Description
`layout`	string	Required. The layout name.
`layoutType`	string	
`sections`	LayoutSectionTranslation[]	An array of layout section translations.

LayoutSectionTranslation

LayoutSectionTranslation contains details for a page layout section translation. For more details, see LayoutSection.

Field	Field Type	Description
`label`	string	Required. Translation for the label. Maximum of 765 characters.
`section`	string	Required. The section name.

LookupFilterTranslation

LookupFilterTranslation shows a translation for a lookup filter error message associated with the custom object. Replaces NamedFilterTranslation.

LookupFilterTranslation is available in API version 30.0 and later.

Field	Field Type	Description
`errorMessage`	string	The error message that appears if the lookup filter fails.
`informationalMessage`	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.

NamedFilterTranslation

NamedFilterTranslation has been removed as of API version 30.0 and is only available in previous API versions.

NamedFilterTranslation shows a list of translations for lookup filter error messages associated with the custom object. See NamedFilter for more information.

Field	Field Type	Description
`errorMessage`	string	The error message that appears if the lookup filter fails.
`informationalMessage`	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.
`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 `name` field.

ObjectNameCaseValue

ObjectNameCaseValue supports multiple cases and definitions of the custom object name to allow usage in various grammatical contexts.

Note: Not every language supports all the possible values for the fields in ObjectNameCaseValue. For language-specific supported values, see the fully supported languages and end-user languages appendices.

Field	Field Type	Description
`article`	Article (enumeration of type string)	English has two types of articles: definite (*`the`*) and indefinite (*`a`*, *`an`*). The usage of these articles depends mainly on whether you are referring to any member of a group, or to a specific member of a group. The valid values are: • `Definite` • `Indefinite` • `None`
`caseType`	CaseType (enumeration of type string)	The case of the custom object name. The valid values are: • `Ablative`

Field	Field Type	Description
		• Accusative • Adessive • Allative • Causalfinal • Dative • Delative • Distributive • Elative • Essive • Essiveformal • Genitive • Illative • Inessive • Instrumental • Lative • Locative • Nominative • Objective • Partitive • Prepositional • Subjective • Sublative • Superessive • Termanative • Translative • Vocative
plural	boolean	Indicates whether the value field is plural (true) or singular (false).
possessive	Possessive (enumeration of type string)	The possessive case of a language is a grammatical case used to indicate a relationship of possession.The valid values are: • First • None • Second
value	string	Required. The value or label in this grammatical context.

PicklistValueTranslation

PicklistValueTranslation contains details for a picklist value translation. For more details, see Picklist (Including Dependent Picklist).

Field	Field Type	Description
`masterLabel`	string	Required. The picklist value defined on the setup page in the application is your master label. The master label is displayed wherever a translated label is not available.
`translation`	string	Required. Translation for the value.

QuickActionTranslation

QuickActionTranslation contains details for an action label in the user interface. For more information, see QuickAction.

Field	Field Type	Description
`label`	string	Required. Translation for the label. Maximum of 765 characters.
`name`	string	Required. The quick action name.

RecordTypeTranslation

RecordTypeTranslation contains details for a record type name translation. For more details, see RecordType.

Field	Field Type	Description
`label`	string	Required. Translation for the label. Maximum of 765 characters.
`name`	string	Required. The record type name.

SharingReasonTranslation

SharingReasonTranslation contains details for a sharing reason translation. For more details, see SharingReason.

Field	Field Type	Description
`label`	string	Required. Translation for the sharing reason.
`name`	string	Required. The sharing reason name.

ValidationRuleTranslation

ValidationRuleTranslation contains details for a validation rule translation. For more details, see ValidationRule.

Field	Field Type	Description
errorMessage	string	Required. Translation for the error message associated with the validation rule failure.
name	string	Required. The validation rule name.

WebLinkTranslation

WebLinkTranslation contains details for a web link translation. For more details, see WebLink.

Field	Field Type	Description
label	string	Required. Translation for the web link label. Maximum of 765 characters.
name	string	Required. The web link name.

WorkflowTaskTranslation

WorkflowTaskTranslation contains details for a workflow task translation. For more details, see Workflow.

Field	Field Type	Description
description	string	Translation for the workflow task description.
name	string	Required. The workflow task name.
subject	string	Translation for the workflow task subject.

Declarative Metadata Sample Definitions

This is a sample XML definition of a CustomObjectTranslation for the Description__c object in German, with one custom field, Summary__c. The name and location of the file containing this definition would be objectTranslations/Description__c-de.objectTranslation.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>false</plural>
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>false</plural>
```

```
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>false</plural>
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>false</plural>
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <fields>
        <label>Zusammenfassung</label>
        <name>Summary__c</name>
    </fields>
    <gender>Feminine</gender>
    <nameFieldLabel>Beschreibungen</nameFieldLabel>
</CustomObjectTranslation>
```

This is a sample XML definition of a CustomObjectTranslation for the Account object, renaming Account to Client (Kunde) in German. The Account object has one standard field, account_number, and one custom field, Account_Code__c. The name and location of the file containing this definition would be `objectTranslations/Account-de.objectTranslation`.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>false</plural>
        <value>Kunde</value>
    </caseValues>
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>false</plural>
        <value>Kunden</value>
```

```
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>false</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>false</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <fields>
        <caseValues>
            <caseType>Nominative</caseType>
            <plural>false</plural>
            <value>Kundennummer</value>
        </caseValues>
        <caseValues>
            <caseType>Nominative</caseType>
            <plural>true</plural>
            <value>Kundennummern</value>
        </caseValues>
        <gender>Feminine</gender>
        <name>account_number</name>
    </fields>
    <fields>
        <label>Kunden-Code</label>
        <name>Account_Code__c</name>
    </fields>
    <gender>Masculine</gender>
</CustomObjectTranslation>
```

SEE ALSO:

CustomObject

Translations

CustomPageWebLink

Represents a web link defined in a home page component. This type extends the Metadata metadata type and inherits its `fullName` field. All other web links are stored as a WebLink in a CustomObject.

Declarative Metadata File Suffix and Directory Location

There is one file per web link definition, stored in the `weblinks` folder in the corresponding package directory. The file suffix is `.weblink`.

Version

CustomPageWebLinks are available in API version 13.0 and later.

Fields

The CustomPageWebLink definition has 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: • `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 it if your template requires data in a different format. This is available if your Content Source is URL.Valid values include: • `UTF-8`—Unicode (UTF-8) • `ISO-8859-1`—General US & Western Europe (ISO-8859–1, ISO-LATIN-1) • `Shift_JIS`—Japanese (Shift-JIS) • `ISO-2022-JP`—Japanese (JIS) • `EUC-JP`—Japanese (EUC-JP) • `x-SJIS_0213`—Japanese (Shift-JIS_2004) • `ks_c_5601-1987`—Korean (ks_c_5601-1987) • `Big5`—Traditional Chinese (Big5) • `GB2312`—Simplified Chinese (GB2312) • `Big5-HKSCS`—Traditional Chinese Hong Kong (Big5–HKSCS)

Field Name	Field Type	Description
`fullName`	string	The name used as a 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 end with an underscore, and not contain two consecutive underscores.
`hasMenubar`	boolean	If the `openType` is `newWindow`, whether to show the browser menu bar for the window (`true` or not (`false`). Otherwise this field should not be specified.
`hasScrollbars`	boolean	If the `openType` is `newWindow`, whether to show the scroll bars for the window (`true`) or not (`false`). Otherwise this field should not be specified.
`hasToolbar`	boolean	If the `openType` is `newWindow`, whether to show the browser toolbar for the window (`true`) or not (`false`). Otherwise this field should not be specified.
`height`	int	Height in pixels of the window opened by this Weblink. Required if the `openType` is `newWindow`, otherwise should not be specified
`isResizable`	boolean	If the `openType` is `newWindow`, whether to allow resizing of the window (`true`) or not (`false`). Otherwise this field should not be specified.
`linkType`	WebLinkType (enumeration of type string)	Required. Represents whether the content of this WebLink is specified by a URL, an sControl, a JavaScript code block, or a Visualforce page. • `url` • `sControl` • `javascript` • `page` • `flow`—Reserved for future use.
`masterLabel`	string	The master label for the Weblink.
`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: • `newWindow` • `sidebar` • `noSidebar` • `replace` • `onClickJavaScript`
`page`	string	If the value of `linkType` is `page`, this field represents the Visualforce page; otherwise, this field should not be specified.

Field Name	Field Type	Description
`position`	WebLinkPosition (enumeration of type string)	If the `openType` is `newWindow`, how the new window should be displayed. Otherwise this field should not be specified. Valid values: • `fullScreen` • `none` • `topLeft`
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`requireRowSelection`	boolean	If the `openType` is `massAction`, whether to require individual row selection to execute the action for this button (`true`) or not (`false`). Otherwise this field should not be specified.
`scontrol`	string	If the value of `linkType` is `sControl`, this field represents the name of the sControl; otherwise, this field should not be specified.
`showsLocation`	boolean	If the `openType` is `newWindow`, whether or not to show the browser location bar for the window; otherwise this field should not be specified.
`showsStatus`	boolean	If the `openType` is `newWindow`, 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 `linkType` is `url`, this is the URL value. If the value of `linkType` is `javascript`, 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 `openType` is `newWindow`, otherwise should not be specified.

Declarative Metadata Sample Definition

The following is the definition of a Weblink. For related samples, see HomePageComponent and HomePageLayout.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomPageWebLink xmlns="http://soap.sforce.com/2006/04/metadata">
    <availability>online</availability>
    <displayType>button</displayType>
    <encodingKey>UTF-8</encodingKey
    <hasMenubar>false</hasMenubar>
    <hasScrollbars>true</hasScrollbars>
    <hasToolbar>false</hasToolbar>
    <height>600</height>
    <isResizable>true</isResizable>
```

```
    <linkType>url</linkType>
    <masterLabel>detailPageButon</masterLabel>
    <openType>newWindow</openType>
    <position>none</position>
    <protected>false</protected>
    <showsLocation>false</showsLocation>
    <showsStatus>false</showsStatus>
    <url>http://google.com</url>
</CustomPageWebLink>
```

SEE ALSO:

- HomePageComponent
- HomePageLayout
- WebLink

CustomPermission

Represents a permission that grants access to a custom feature. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CustomPermission components have the suffix `.customPermission` and are stored in the `customPermissions` folder.

Version

CustomPermission components are available in API version 31.0 and later.

Fields

Field Name	Field Type	Description
`connectedApp`	string	The name of the connected app that's associated with this permission. Limit: 80 characters.
`description`	string	The custom permission description. Limit: 255 characters.
`label`	string	Required. The custom permission label. Limit: 80 characters.
`requiredPermission`	CustomPermissionDependencyRequired[]	Indicates which custom permissions are required by the parent custom permission. This field is available in API version 32.0 and later.

CustomPermissionDependencyRequired

CustomPermissionDependencyRequired determines whether a custom permission is required by the parent custom permission. A required custom permission must be enabled when its parent is enabled.

Field Name	Field Type	Description
`customPermission`	string	Required. The custom permission name.
`dependency`	boolean	Required. Indicates whether this custom permission is required by the parent custom permission (`true`) or not (`false`).

Declarative Metadata Sample Definition

The following is an example of a CustomPermission component.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomPermission xmlns="http://soap.sforce.com/2006/04/metadata">
   <connectedApp>Acme</connectedApp>
   <description>Read and edit access for Acme accounts.</description>
   <label>Acme Account Full Access</label>
   <requiredPermission>
      <customPermission>Acme_Account_Read</customPermission>
      <dependency>true</dependency>
   </requiredPermission>
</CustomPermission>
```

The following is an example `package.xml` that references the previous definition, as well as other custom permissions that are associated with a connected app.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>Acme</members>
      <name>ConnectedApp</name>
   </types>
   <types>
      <members>Acme_Account_Email_Read</members>
      <members>Acme_Account_Phone_Edit</members>
      <members>Acme_Account_Full_Access</members>
      <members>Acme_Account_Read</members>
      <name>CustomPermission</name>
   </types>
   <types>
      <members>Acme_Account_Email_Read</members>
      <members>Acme_Account_Phone_Edit</members>
      <members>Acme_Account_Full_Access</members>
      <members>Acme_Account_Read</members>
      <name>PermissionSet</name>
   </types>
   <version>36.0</version>
</Package>
```

CustomSite

Represents a Force.com site. Force.com Sites enables you to create public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password. For more information, see "Force.com Sites Overview" in the Salesforce online help.

Note: CustomSite does not support syndication feeds at this time.

This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

Force.com CustomSite components are stored in the `sites` directory of the corresponding package directory. The file name matches the site name, and the extension is `.site`.

Version

Force.com CustomSite components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`active`	boolean	Required. Determines whether or not the site is active.
`allowHomePage`	boolean	Determines whether or not the standard home page is visible to public users. This is a new field in API version 15.0.
`allowStandardAnswersPages`	boolean	Determines whether the standard answer pages are visible to public users. This is a new field in API version 19.0.
`allowStandardIdeasPages`	boolean	Determines whether or not the standard Ideas pages are visible to public users. This is a new field in API version 15.0.
`allowStandardLookups`	boolean	Determines whether or not the standard lookup pages are visible to public users. This is a new field in API version 15.0.
`allowStandardSearch`	boolean	Determines whether or not the standard search pages are visible to public users. This is a new field in API version 15.0.
`analyticsTrackingCode`	string	The tracking code associated with your site. This code can be used by services like Google Analytics to track page request data for your site. This field is available in API version 17.0 and later.

Field	Field Type	Description
`authorizationRequiredPage`	string	The name of the Visualforce page to be displayed when the guest user tries to access a page for which they are not authorized.
`bandwidthExceededPage`	string	The name of the Visualforce page to be displayed when the site has exceeded its bandwidth quota.
`changePasswordPage`	string	The name of the Visualforce page to be displayed when the portal user attempts to change his or her password for either the portal or for Chatter Answers, when enabled.
`chatterAnswersForgotPasswordConfirmPage`	string	The name of the Visualforce page to be displayed that informs the user that an email has been sent to them with a temporary password. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersForgotPasswordPage`	string	The name of the Visualforce page to be displayed when a user clicks the link to retrieve a forgotten password. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersHelpPage`	string	The name of the Visualforce page to be displayed when the user clicks the help link. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersLoginPage`	string	The name of the Visualforce page to be displayed to allow users to log in to the portal. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersRegistrationPage`	string	The name of the Visualforce page to be displayed to allow users to register themselves and access the portal. This field is available in API version 27.0 and later.
`clickjackProtectionLevel`	SiteClickjackProtectionLevel (enumeration of type string)	Required. Sets the clickjack protection level. The options are: • `AllowAllFraming` — Allow framing by any page (no protection) • `SameOriginOnly` — Allow framing by the same origin only (recommended) • `NoFraming` — Don't allow framing by any page (most protection) This field is available in API version 30.0 and later.

Field	Field Type	Description
customWebAddresses	SiteWebAddress[]	The root custom URLs associated with the site. Saving or deploying a CustomSite replaces all root custom URLs in the site with the root custom URLs in this list. Custom URLs that use a non-root path prefix are not included in this list and are not affected when saving or deploying a CustomSite. This field is available in API version 21.0 and later.
description	string	The site description.
favoriteIcon	string	The name of the file to be used for the icon that appears in the browser's address field when visiting the site. Sets the favorite icon for the entire site.
fileNotFoundPage	string	The name of the Visualforce page to be displayed when the guest user tries to access a non-existent page.
forgotPasswordPage	string	The name of the Visualforce page to be displayed when a user clicks the Forgot Password link on the site's login page. This field is only applicable for Communities sites.
genericErrorPage	string	The name of the Visualforce page to be displayed for errors not otherwise specified.
guestProfile	string	Read only. The name of the profile associated with the guest user.
inMaintenancePage	string	The name of the Visualforce page to be displayed when the site is down for maintenance.
inactiveIndexPage	string	The name of the Visualforce page set as the inactive site home page.
indexPage	string	Required. The name of the Visualforce page set as the active site home page.
masterLabel	string	The name of the site label in the Salesforce user interface.
portal	string	The name of the portal associated with this site for login access.
requireHttps	boolean	Determines whether the site requires secure connections (true) or not (false). When false, the site operates normally via insecure connections instead of redirecting to a secure connection.
requireInsecurePortalAccess	boolean	Required. Determines whether to override your organization's security settings and exclusively use HTTP when logging in to the associated portal from your site.

Field	Field Type	Description
`robotsTxtPage`	string	The name of the Visualforce page to be displayed for the `robots.txt` file used by web crawlers.
`serverIsDown`	string	The name of the static resource to be displayed from the cache server when Salesforce servers are down. The static resource must be a public zip file 1 MB or smaller and must contain a page named `maintenance.html` at the root level of the zip file. Other resources in the zip file, such as images or CSS files, can follow any directory structure. This field is available in API version 17.0 and later.
`siteRedirectMappings`	SiteRedirectMapping[]	An array of all URL redirect rules set for your site. This field is available in API version 20.0 and later.
`siteAdmin`	string	The username of the site administrator.
`siteTemplate`	string	The name of the Visualforce page to be used as the site template.
`siteType`	siteType	Identifies whether the site is a Visualforce (Force.com Sites) or a Site.com site. If Salesforce Communities is enabled for your organization, the site could also be a ChatterNetwork (Force.com Sites) or ChatterNetworkPicasso (Site.com) site. This is a new field in API version 27.0.
`subdomain`	string	Required. Read only. The custom subdomain prefix for the site. For example, if your site URL is `mycompany.force.com/partners`, `mycompany.force.com` is the subdomain.
`urlPathPrefix`	string	The first part of the path on the site's URL that distinguishes this site from other sites. For example, if your site URL is `mycompany.force.com/partners`, `partners` is the urlPathPrefix.

SiteRedirectMapping

SiteRedirectMapping represents a URL redirect rule on your Force.com site. For more information, see "Force.com Sites URL Redirects" in the Salesforce online help.

Field	Field Type	Description
`action`	SiteRedirect (enumeration of type string)	The type of the redirect. Available string values are: • Permanent

Field	Field Type	Description
		• Temporary
`isActive`	boolean	The status of the redirect: active or inactive.
`source`	string	The URL that you want to redirect. It must be a relative URL, but can have any valid extension type, such as `.html` or `.php`.
`target`	string	The new URL you want users to visit. It can be a relative URL or a fully-qualified URL with an `http://` or `https://` prefix.

SiteWebAddress

Represents the web address of a Force.com site.

Field	Field Type	Description
`certificate`	string	Although this field is visible in the Metadata API version 36.0, it's not functional and should be left blank.
`domainName`	string	The domain of the website, in the form of www.acme.com.
`primary`	boolean	Indicates whether this is the primary domain (*`true`*). If *`false`*, this is not the primary domain.

Declarative Metadata Sample Definition

A sample XML definition of a site is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomSite xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <allowHomePage>true</allowHomePage>
    <allowStandardIdeasPages>true</allowStandardIdeasPages>
    <allowStandardLookups>true</allowStandardLookups>
    <allowStandardSearch>true</allowStandardSearch>
    <authorizationRequiredPage>Unauthorized</authorizationRequiredPage>
    <bandwidthExceededPage>BandwidthExceeded</bandwidthExceededPage>
    <changePasswordPage>ChangePassword</changePasswordPage>

<chatterAnswersForgotPasswordConfirmPage>ChatterAnswersForgotPasswordConfirm</chatterAnswersForgotPasswordConfirmPage>


<chatterAnswersForgotPasswordPage>ChatterAnswersForgotPassword</chatterAnswersForgotPasswordPage>

    <chatterAnswersHelpPage>ChatterAnswersHelp</chatterAnswersHelpPage>
```

```
    <chatterAnswersLoginPage>ChatterAnswersLogin</chatterAnswersLoginPage>

<chatterAnswersRegistrationPage>ChatterAnswersRegistration</chatterAnswersRegistrationPage>

    <clickjackProtectionLevel>SameOriginOnly</clickjackProtectionLevel>
    <customWebAddresses>
      <domainName>www.testing123.com</domainName>
      <primary>true</primary>
    </customWebAddress>
    <favoriteIcon>myFavIcon</favoriteIcon>
    <fileNotFoundPage>FileNotFound</fileNotFoundPage>
    <genericErrorPage>Exception</genericErrorPage>
    <inMaintenancePage>InMaintenance</inMaintenancePage>
    <serverIsDown>MyServerDownResource</serverIsDown>
    <indexPage>UnderConstruction</indexPage>
    <masterLabel>customSite</masterLabel>
    <portal>Customer Portal</portal>
    <requireInsecurePortalAccess>false</requireInsecurePortalAccess>
    <siteAdmin>admin@myco.org</siteAdmin>
    <siteTemplate>SiteTemplate</siteTemplate>
    <subdomain>myco</subdomain>
</CustomSite>
```

SEE ALSO:

Portal

CustomTab

Represents a custom tab. Display custom object data or other Web content using custom tabs in Salesforce. When a tab displays a custom object, the tab name is the same as the custom object name; for page, s-control, or URL tabs, the name is arbitrary. For more information, see "Understanding Custom Tabs" in the Salesforce online help. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

The file suffix is `.tab`. There is one file for each tab, stored in the `tabs` 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

Tabs are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`auraComponent`	string	Indicates whether this tab is for a Lightning component (`true`) or not (`false`). If set to `true`, the name of the tab matches the name of the Lightning component. Only one of these fields should have a value set: • `auraComponent` • `customObject` • `flexiPage` • `page` • `scontrol` • `url` Note: `auraComponent` is part of the Lightning Components (Beta) feature.
`customObject`	boolean	Indicates whether this tab is for a custom object (`true`) or not (`false`). If set to `true`, the name of the tab matches the name of the custom object. Only one of these fields should have a value set: • `auraComponent` • `customObject` • `flexiPage` • `page` • `scontrol` • `url`
`description`	string	The optional description text for the tab.
`flexiPage`	string	The name of the Lightning Page to display in this tab. Only one of these fields should have a value set: • `auraComponent` • `customObject` • `flexiPage` • `page` • `scontrol` • `url`
`frameHeight`	int	The height, in pixels of the tab frame. Required for s-control and page tabs.
`fullName`	string	The name of the tab. The value of this field depends on the type of tab, and the API version. • For custom object tabs, the `fullName` is the developer-assigned name of the custom object (MyCustomObject__c, for example). For

Field Name	Field Type	Description
		custom object tabs, this name must be the same as the custom object name, and `customObject` should be set to `true`. • For Web tabs, the `fullName` is the developer-assigned name of the tab (MyWebTab, for example). The `fullName` 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.
`hasSidebar`	boolean	Indicates if the tab displays the sidebar panel.
`icon`	string	The optional reference to the image document for the tab if the tab is not using one of the standard tab styles. This is a new field in API version 14.0.
`label`	string	This is the label of the tab, for Web tabs only.
`mobileReady`	boolean	Required. Indicates if the custom tab is available for Mobile Edition (`true`) or not (`false`).
`motif`	string	Required. The tab style for the color scheme and icon for the custom tab. For example, "'Custom70: Handsaw," is the handsaw icon.
`page`	string	The name of the Visualforce page to display in this tab. Only one of these fields should have a value set: • `auraComponent` • `customObject` • `flexiPage` • `page` • `scontrol` • `url`
`scontrol`	string	The name of the s-control to display in this tab. Only one of these fields should have a value set: • `auraComponent` • `customObject` • `flexiPage` • `page` • `scontrol` • `url`
`splashPageLink`	string	The custom link used as the introductory splash page when users click the tab. References a HomePageComponent.

Field Name	Field Type	Description
`url`	string	The URL for the external web-page to embed in this tab. Only one of these fields should have a value set: • `auraComponent` • `customObject` • `flexiPage` • `page` • `scontrol` • `url`
`urlEncodingKey`	Encoding (enumeration of type string)	The default encoding setting is Unicode: `UTF-8`. Change it if you are passing information to a URL that requires data in a different format. This option is available when the value `URL` is selected in the tab type.

Declarative Metadata Sample Definition

The following is the definition of a tab:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomTab xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Myriad Publishing</description>
    <frameHeight>600</frameHeight>
    <mobileReady>true</mobileReady>
    <motif>Custom53: Bell</motif>
    <url>http://www.myriadpubs.com</url>
    <urlEncodingKey>UTF-8</urlEncodingKey>
</CustomTab>
```

SEE ALSO:

CustomApplication

Dashboard

Represents a dashboard. Dashboards are visual representations of data that allow you to see key metrics and performance at a glance. This type extends the Metadata metadata type and inherits its `fullName` field. For more information, see "Edit Dashboards in Accessibility Mode" in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Dashboards are stored in the `dashboards` directory of the corresponding package directory. The file name matches the dashboard title and the extension is `.dashboard`.

Retrieving Dashboards

You can't use the wildcard (*) symbol with dashboards in `package.xml`. To retrieve the list of dashboards for populating `package.xml` with explicit names, call `listMetadata()` and pass in `DashboardFolder` as the type. Note that DashboardFolder is not returned as a type in `describeMetadata()`. Dashboard is returned from `describeMetadata()` with an associated attribute of `inFolder` set to true. If that attribute is set to true, you can construct the type by using the component name with the word Folder, such as DashboardFolder.

The following example shows folders in `package.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
        <name>Report</name>
    </types>
    <version>36.0</version>
</Package>
```

Version

Dashboard components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`backgroundEndColor`	string	Required. A dashboard can have a gradient color change on its charts. This field defines the second color for the gradient and `backgroundStartColor` defines the first color. If you prefer your background to be all one color or do not want a gradient color change, select the same color for this field and `backgroundStartColor`. The color is in hexadecimal format; for example #FF6600.

Field	Field Type	Description
backgroundFadeDirection	ChartBackgroundDirection (enumeration of type string)	Required. The direction of the gradient color change, defined by the backgroundStartColor and backgroundEndColor fields. The valid values are: • diagonal • leftToRight • topToBottom
backgroundStartColor	string	Required. The starting color for the gradient color change on the dashboard's charts. See backgroundEndColor for more information. The color is in hexadecimal format; for example #FF6600.
dashboardFilters	DashboardFilters[]	The list of filters in a dashboard. This field is available in API version 23.0 and later.
dashboardGridLayout	DashboardGridLayout	Lists the included DashboardGridComponent objects, specifies the number of dashboard columns, and sets each dashboard row's height in pixels. This field is available in API version 35.0 and later.
dashboardType	DashboardType (enumeration of type string)	Determines the way visibility settings are set for a dashboard. The valid values are: • SpecifiedUser—All users see data at the access level of one specific running user, specified in the runningUser field, regardless of their own security settings. • LoggedInUser—Each logged-in user sees data according to his or her own access level. • MyTeamUser—Managers can choose to view the dashboard from the point of view of their subordinates in the role hierarchy. This value is available in API version 20.0 and later. This field is available in API version 19.0 and later.
description	string	Description for the dashboard. Maximum of 255 characters.
folderName	string	Name of the folder that houses the dashboard. This field is available in API version 35.0 and later.
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 create() to see an example of this field specified for a call. This field specifies the folder and dashboard title; for example folderSales/California.

Field	Field Type	Description
`isGridLayout`	boolean	Specifies whether a dashboard uses the Lightning Experience layout (`true`) or not (`false`). Lightning Experience allows dashboards with more than three columns with components that span multiple columns and multiple rows in size. This field is available in API version 35.0 and later.
`leftSection`	DashboardComponentSection	Required. The left section or column of the dashboard.
`middleSection`	DashboardComponentSection	The middle section or column of the dashboard.
`rightSection`	DashboardComponentSection	Required. The right section or column of the dashboard.
`runningUser`	string	The username of the user whose role and sharing settings are used to determine the data shown in the dashboard. When you deploy a dashboard and the value in this field is not defined or does not correspond to a valid user, the field is populated with the username of the user performing the deployment. Regardless of their security settings, all users viewing a dashboard see exactly the same data, because dashboards are always run using the security settings of a particular user. Tip: To avoid inappropriate exposure of sensitive data, save the dashboard to a folder that is visible only to appropriate users.
`textColor`	string	Required. Color of the text on each chart in the dashboard. The color is in hexadecimal format; for example #FF6600.
`title`	string	Required. The dashboard title.
`titleColor`	string	Required. Color of the titles on each dashboard component. The color is in hexadecimal format; for example #FF6600.
`titleSize`	int	Required. Size of characters in title text. For example, a value of 12 indicates 12pt text.

DashboardComponentSection

DashboardComponentSection represents one of the sections or columns in a dashboard.

Field	Field Type	Description
`columnSize`	DashboardComponentSize (enumeration of type string)	Required. The size of the column in the dashboard. See DashboardComponentSize for details on valid values.
`components`	DashboardComponent[]	The list of DashboardComponent objects in the dashboard column.

DashboardComponentSize

DashboardComponentSize is an enumeration of type string that lists different size categories. The valid values are listed in the table below:

Enumeration Value	Description
`medium`	Medium component size.
`narrow`	Smallest component size.
`wide`	Largest component size.

DashboardComponent

A dashboard consists of a group of different components or elements that display data. Each component can use a custom report or a custom s-control as their data source to display corporate metrics or key performance indicators. You can create several dashboard components and display them all in one dashboard aligned in up to three columns.

Field	Field Type	Description
`chartAxisRange`	ChartRangeType (enumeration of type string)	A manual or automatic axis range for bar or line charts. The valid values are: • `auto` • `manual`
`chartAxisRangeMax`	double	The maximum axis range to be displayed. This only applies to bar and line charts in which the `manual` axis range is selected for the `chartAxisRange` field.
`chartAxisRangeMin`	double	The minimum axis range to be displayed. This only applies to bar and line charts in which the `manual` axis range is selected for the `chartAxisRange` field.
`chartSummary`	ChartSummary	Specifies the summary field for the chart data. Required if `isAutoSelectFromReport` is set to `false`. This field is available in API version 25.0 and later.
`componentType`	DashboardComponentType (enumeration of type string)	Required. Dashboard component type. The valid values are: • `Bar` • `BarGrouped` • `BarStacked` • `BarStacked100` • `Column` • `ColumnGrouped` • `ColumnLine` • `ColumnLineGrouped`

Field	Field Type	Description
		• ColumnLineStacked • ColumnLineStacked100 • ColumnStacked • ColumnStacked100 • Donut • Funnel • Gauge • Line • lineCumulative • LineGrouped • lineGroupedCumulative • Metric • Pie • Scatter • ScatterGrouped • Scontrol • Table
dashboardFilterColumns	DashboardFilterColumns[]	A list of dashboard filter columns. Each report-based component must have a dashboard filter column that defines the column that the filter applies to. This field is available in API version 23.0 and later.
dashboardTableColumn	DashboardTableColumn[]	Represents a list of columns on a customized dashboard table component.
displayUnits	ChartUnits (enumeration of type string)	Chart Units. The valid values are: • Auto • Integer • Hundreds • Thousands • Millions • Billions • Trillions
drillDownUrl	string	For charts, specifies a URL that users go to when they click the dashboard component. Use this option to send users to another dashboard, report, record detail page, or other system that uses a Web interface. This field overrides the drillEnabled and drillToDetailEnabled fields.

Field	Field Type	Description
`drillEnabled`	boolean	Specifies whether to take users to the full or filtered source report when they click the dashboard component. Set to `false` to drill to the full source report; set to `true` to drill to the source report filtered by what they clicked. If set to `true`, users can click individual groups, axis values, or legend entries. This overrides the `drillToDetailEnabled` field. This field is available in API version 17.0 and later.
`drillToDetailEnabled`	boolean	When enabled, users are taken to the record detail page when they click a record name, record owner, or feed post in a table or chart. When set to `true` users can click axis and legend values, chart elements, and table entries. The `drillDownUrl` and `drillEnabled` fields override this field. This field is available in API version 20.0 and later.
`enableHover`	boolean	Specifies whether to display values, labels, and percentages when hovering over charts. Hover details depend on chart type. Percentages apply to pie, donut, and funnel charts only. This field is available in API version 17.0 and later.
`expandOthers`	boolean	Specifies whether to combine all groups less than or equal to 3% of the total into a single 'Others' wedge or segment. This only applies to pie, donut, and funnel charts. Set to `true` to show all values individually on the chart; set to `false` to combine small groups into 'Others.' This field is available in API version 17.0 and later.
`footer`	string	Footer displayed at the bottom of the dashboard component. Maximum of 255 characters.
`gaugeMax`	double	The maximum value on a gauge. A gauge is used to see how far you are from reaching a goal. It looks like a speedometer in a car.
`gaugeMin`	double	The minimum value on a gauge.
`groupingColumn`	string	Specifies the field by which to group data. This data is displayed on the X-axis for vertical column charts and on the Y-axis for horizontal bar charts. This field is available in API version 25.0 and later.
`header`	string	Header displayed at the top of the dashboard component. Maximum of 80 characters.
`indicatorBreakpoint1`	double	The value that separates the `indicatorLowColor` from the `indicatorMiddleColor` on the dashboard.

Field	Field Type	Description
`indicatorBreakpoint2`	double	The value that separates the `indicatorMiddleColor` from the `indicatorHighColor` on the dashboard.
`indicatorHighColor`	string	The color representing a high number range on the gauge.
`indicatorLowColor`	string	The color representing a low number range on the gauge.
`indicatorMiddleColor`	string	The color representing a medium number range on the gauge.
`legendPosition`	ChartLegendPosition (enumeration of type string)	The location of the legend with respect to the chart. The valid values are: • `Bottom` • `OnChart` • `Right`
`maxValuesDisplayed`	int	The maximum number of elements to include in the top-level grouping of the horizontal axis of a horizontal chart, vertical axis of a vertical chart, or selected axis of a stacked bar chart. For example, if you want to list only your top five salespeople, create an opportunity report that lists total opportunity amounts by owner and enter `5` in this field.
`metricLabel`	string	Descriptive label for the metric. This is relevant if `metric` is the value of the `componentType` field.
`page`	string	Visualforce page associated with the component.
`pageHeightInPixels`	int	Display height of the Visualforce page in pixels.
`report`	string	Name of the report associated with the component.
`scontrol`	string	S-control associated with component if `scontrol` is the value of the `componentType` field. For more information, see "Defining Custom S-Controls" in the Salesforce online help.
`scontrolHeightInPixels`	int	Display height of the s-control in pixels.
`showPercentage`	boolean	Indicates if percentages are displayed for regions of gauges and wedges and segments of pie, donut, and funnel charts (`true`), or not (`false`).
`showPicturesOnCharts`	boolean	Display Chatter photos for up to 20 records in a horizontal bar chart component whose source report is grouped by a user or group name field. If there are more than 20 records with photos, record names are shown instead of photos. Set `Grouping Display` to *`None`* to show

Field	Field Type	Description
		photos. Set the `Drill Down to` option to *`Record Detail Page`* to take users directly to user profile or group pages when they click photos. Chatter must be enabled for photos to be displayed. Depending on your organization's setup, you may not see photos on tables and charts.
`showPicturesOnTables`	boolean	Display Chatter photos for up to 20 records in a horizontal bar chart component whose source report is grouped by a user or group name field. If there are more than 20 records with photos, record names are shown instead of photos. Set `Grouping Display` to *`None`* to show photos. Set the `Drill Down to` option to *`Record Detail Page`* to take users directly to user profile or group pages when they click photos. Chatter must be enabled for photos to be displayed. Depending on your organization's setup, you may not see photos on tables and charts.
`showTotal`	boolean	Indicates if the total of all wedges is displayed for gauges and donut charts (`true`), or not (`false`).
`showValues`	boolean	Indicates if the values of individual records or groups are displayed for charts (`true`), or not (`false`).
`sortBy`	DashboardComponentFilter (enumeration of type string)	The sort option for the dashboard component.
`title`	string	The title of the dashboard component. Maximum of 40 characters.
`useReportChart`	boolean	Specifies whether to use the chart defined in the source report on this dashboard component. The chart settings in the source report determine how the chart displays in the dashboard, and any chart settings you define for the dashboard are overridden. If you defined a combination chart in the source report, use this option to use that combination chart on this dashboard.

DashboardFilters

DashboardFilters represents a filter in a dashboard.

Field	Field Type	Description
`dashboardFilterOptions`	DashboardFilterOptions[]	The list of items you can select in the `Filter Options` section of the Add Filter dialog.
`name`	string	Required. The filter label.

DashboardFilterColumns

DashboardFilterColumns represents a filter column in a dashboard.

Field	Field Type	Description
`column`	string	Required. The report column code for the filter.

DashboardFilterOptions

DashboardFilterOptions represents a filter option in a dashboard.

Field	Field Type	Description
`operator`	DashboardFilterOperation (enumeration of type string)	Required. Represents the filter operation for this filter item. Valid values are enumerated in DashboardFilterOperation. This field is available in API version 24.0 and later. With API version 23.0, valid values are enumerated in FilterOperation.
`value`	string	Required. The value in the `Filter Options` area of the `Add Filter` dialog. This field is available in API version 23.0.
`values`	string[]	Required. One or more values in the `Filter Options` area of the `Add Filter` dialog. This field is available in API version 24.0 and later.

DashboardFilterOperation

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

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

Note: The "between" operator takes two operands (for example, "between MinimumValue, MaximumValue"). Note also that the minimum value is inclusive, while the maximum value is exclusive. All other dashboard filter operations take a single operand only.

DashboardTableColumn

DashboardTableColumn represents a column in a customized table component in a dashboard.

Field	Field Type	Description
`aggregateType`	ReportSummaryType[] (enumeration of type string)	Specifies the aggregation type for the table column.
`column`	string	Required. The label of the column to use in the table.
`showTotal`	boolean	Displays the totals for each summarizable column in the dashboard table. This field is available in API version 19.0 and later.
`sortBy`	DashboardComponentFilter (enumeration of type string)	The sort option for the dashboard table component. Sort on just one column per table.

DashboardComponentFilter

DashboardComponentFilter is an enumeration of type string that lists the sort values for dashboard components. The valid values are:

Enumeration Value	Description
`RowLabelAscending`	Sorts in alphabetical order by the label.
`RowLabelDescending`	Sorts in reverse alphabetical order by the label.
`RowValueAscending`	Sorts lowest to highest by the value.
`RowValueDescending`	Sorts highest to lowest by the value.

DashboardGridComponent

Lightning Experience features dashboards with more than three columns and components that span multiple columns and multiple rows in size. DashboardGridComponent specifies location and size of a given dashboard component.

Field	Field Type	Description
`colSpan`	int	Required. The width of the dashboard component in columns. For example, if `colSpan` is 5, then the dashboard component spans five columns.
`columnIndex`	int	Required. The left-most column that is occupied by the dashboard component.
`dashboardComponent`	DashboardComponent	Required. The dashboard component that is being sized and placed.
`rowIndex`	int	Required. The top-most row that is occupied by the dashboard component.

Field	Field Type	Description
rowSpan	int	Required. The height of the dashboard component in rows.

DashboardGridLayout

Lightning Experience features dashboards with more than three columns and components that span multiple columns and multiple rows in size. DashboardGridLayout lists the included dashboard components, specifies the number of dashboard columns, and sets each dashboard row's height in pixels.

Field	Field Type	Description
dashboardGridComponents	DashboardGridComponent[]	List of DashboardGridComponent objects in the dashboard.
numberOfColumns	int	Required. Total number of columns in the dashboard.
rowHeight	int	Required. Height of each row in pixels.

Declarative Metadata Sample Definition — Filtered Dashboard

A sample XML definition of a filtered dashboard is shown below. Note that this example is supported in API version 24.0 and later. The file name matches the dashboard title and the extension is `.dashboard`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Dashboard xmlns="http://soap.sforce.com/2006/04/metadata">
    <backgroundEndColor>#FFFFFF</backgroundEndColor>
    <backgroundFadeDirection>Diagonal</backgroundFadeDirection>
    <backgroundStartColor>#FFFFFF</backgroundStartColor>
    <dashboardFilters>
        <dashboardFilterOptions>
            <operator>equals</operator>
            <values>Media</values>
        </dashboardFilterOptions>
        <dashboardFilterOptions>
            <operator>lessThan</operator>
            <values>Working</values>
        </dashboardFilterOptions>
        <dashboardFilterOptions>
            <operator>between</operator>
            <values>ABC</values>
            <values>XYZ</values>
        </dashboardFilterOptions>
        <name>Industry</name>
    </dashboardFilters>
    <dashboardFilters>
        <dashboardFilterOptions>
            <operator>equals</operator>
            <values>Analyst,Partner</values>
        </dashboardFilterOptions>
        <dashboardFilterOptions>
            <operator>startsWith</operator>
            <values>Integrator</values>
```

```
        </dashboardFilterOptions>
        <name>Account Type</name>
    </dashboardFilters>
    <dashboardType>SpecifiedUser</dashboardType>
    <leftSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Bar</componentType>
            <dashboardFilterColumns>
                <column>INDUSTRY</column>
            </dashboardFilterColumns>
            <dashboardFilterColumns>
                <column>TYPE</column>
            </dashboardFilterColumns>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/SampleReportofAccounts</report>
            <showPercentage>false</showPercentage>
            <showPicturesOnCharts>false</showPicturesOnCharts>
            <showValues>false</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </components>
    </leftSection>
    <middleSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Funnel</componentType>
            <dashboardFilterColumns>
                <column>ACCOUNT_INDUSTRY</column>
            </dashboardFilterColumns>
            <dashboardFilterColumns>
                <column>ACCOUNT.TYPE</column>
            </dashboardFilterColumns>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/SampleReportofCases</report>
            <showPercentage>false</showPercentage>
            <showValues>true</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </components>
    </middleSection>
    <rightSection>
```

```
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Column</componentType>
            <dashboardFilterColumns>
                <column>INDUSTRY</column>
            </dashboardFilterColumns>
            <dashboardFilterColumns>
                <column>ACCOUNT_TYPE</column>
            </dashboardFilterColumns>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/SampleReportofOpportunities</report>
            <showPercentage>false</showPercentage>
            <showValues>false</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </components>
    </rightSection>
    <runningUser>admin@TESTORGNUM</runningUser>
    <textColor>#000000</textColor>
    <title>My Dashboard</title>
    <titleColor>#000000</titleColor>
    <titleSize>12</titleSize>
</Dashboard>
```

Declarative Metadata Sample Definition — Unfiltered Dashboard

A sample XML definition of a dashboard is shown below. The file name matches the dashboard title and the extension is `.dashboard`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Dashboard xmlns="http://soap.sforce.com/2006/04/metadata">
    <backgroundEndColor>#FFFFFF</backgroundEndColor>
    <backgroundFadeDirection>LeftToRight</backgroundFadeDirection>
    <backgroundStartColor>#FFFFFF</backgroundStartColor>
    <description>Dashboard with all possible chart types</description>
    <leftSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>BarStacked100</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <componentType>Table</componentType>
            <dashboardTableColumn>
```

```
            <column>CLOSE_DATE</column>
            <sortBy>RowLabelAscending</sortBy>
        </dashboardTableColumn>
        <dashboardTableColumn>
            <aggregateType>Sum</aggregateType>
            <column>AMOUNT</column>
            <showTotal>true</showTotal>
        </dashboardTableColumn>
        <dashboardTableColumn>
            <column>STAGE_NAME</column>
        </dashboardTableColumn>
        <dashboardTableColumn>
            <column>PROBABILITY</column>
            <aggregateType>Maximum</aggregateType>
        </dashboardTableColumn>
        <displayUnits>Integer</displayUnits>
        <header>Opportunities Table</header>
        <indicatorHighColor>#54C254</indicatorHighColor>
        <indicatorLowColor>#C25454</indicatorLowColor>
        <indicatorMiddleColor>#C2C254</indicatorMiddleColor>
        <maxValuesDisplayed>10</maxValuesDisplayed>
        <report>testFolder/sourceRep</report>
    </components>
    <components>
        <chartAxisRange>Auto</chartAxisRange>
        <componentType>Bar</componentType>
        <displayUnits>Auto</displayUnits>
        <drillEnabled>true</drillEnabled>
        <enableHover>true</enableHover>
        <report>testFolder/sourceRep</report>
        <sortBy>RowLabelAscending</sortBy>
    </components>
    <components>
        <chartAxisRange>Auto</chartAxisRange>
        <componentType>Column</componentType>
        <displayUnits>Auto</displayUnits>
        <drillEnabled>true</drillEnabled>
        <legendPosition>Bottom</legendPosition>
        <report>testFolder/sourceRep</report>
        <sortBy>RowLabelAscending</sortBy>
        <useReportChart>true</useReportChart>
    </components>
    <components>
        <chartAxisRange>Auto</chartAxisRange>
        <componentType>Funnel</componentType>
        <displayUnits>Auto</displayUnits>
        <drillEnabled>true</drillEnabled>
        <enableHover>true</enableHover>
        <expandOthers>true</expandOthers>
        <legendPosition>Bottom</legendPosition>
        <report>testFolder/sourceRep</report>
        <sortBy>RowLabelAscending</sortBy>
    </components>
</leftSection>
```

```
    <middleSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnStacked100</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnStacked</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnStacked</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnGrouped</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Column</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
    </middleSection>
    <rightSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Bar</componentType>
            <displayUnits>Auto</displayUnits>
```

```
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Pie</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <expandOthers>true</expandOthers>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>LineGroupedCumulative</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>LineGrouped</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>LineCumulative</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Donut</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <expandOthers>true</expandOthers>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
    </rightSection>
    <runningUser>admin@TESTORGNUM</runningUser>
```

```
    <textColor>#000000</textColor>
    <title>Db Title</title>
    <titleColor>#000000</titleColor>
    <titleSize>12</titleSize>
</Dashboard>
```

Declarative Metadata Sample Definition — Lightning Experience Dashboard with `isGridLayout` Equals `true`

A sample XML definition of a Lightning Experience dashboard with `isGridLayout` equals `true` is shown below. Note that this example is supported in API version 35.0 and later. The file name matches the dashboard title and the extension is `.dashboard`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Dashboard xmlns="http://soap.sforce.com/2006/04/metadata">
    <backgroundEndColor>#FFFFFF</backgroundEndColor>
    <backgroundFadeDirection>Diagonal</backgroundFadeDirection>
    <backgroundStartColor>#FFFFFF</backgroundStartColor>
    <dashboardType>SpecifiedUser</dashboardType>
    <gridLayout>
        <dashboardGridComponents>
            <colSpan>3</colSpan>
            <columnIndex>0</columnIndex>
            <dashboardComponent>
                <autoselectColumnsFromReport>false</autoselectColumnsFromReport>
                <chartAxisRange>Auto</chartAxisRange>
                <chartSummary>
                    <axisBinding>y</axisBinding>
                    <column>RowCount</column>
                </chartSummary>
                <componentType>Donut</componentType>
                <drillEnabled>false</drillEnabled>
                <drillToDetailEnabled>false</drillToDetailEnabled>
                <enableHover>false</enableHover>
                <expandOthers>false</expandOthers>
                <groupingColumn>TITLE</groupingColumn>
                <legendPosition>Bottom</legendPosition>
                <report>unfiled$public/lead_rpt</report>
                <showPercentage>false</showPercentage>
                <showTotal>false</showTotal>
                <showValues>true</showValues>
                <sortBy>RowLabelAscending</sortBy>
                <useReportChart>false</useReportChart>
            </dashboardComponent>
            <rowIndex>0</rowIndex>
            <rowSpan>3</rowSpan>
        </dashboardGridComponents>
        <dashboardGridComponents>
            <colSpan>3</colSpan>
            <columnIndex>0</columnIndex>
            <dashboardComponent>
                <autoselectColumnsFromReport>false</autoselectColumnsFromReport>
                <chartAxisRange>Auto</chartAxisRange>
                <chartSummary>
```

```
                <axisBinding>y</axisBinding>
                <column>RowCount</column>
            </chartSummary>
            <componentType>Pie</componentType>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <groupingColumn>TITLE</groupingColumn>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/lead_rpt</report>
            <showPercentage>false</showPercentage>
            <showValues>true</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </dashboardComponent>
        <rowIndex>3</rowIndex>
        <rowSpan>3</rowSpan>
    </dashboardGridComponents>
    <dashboardGridComponents>
        <colSpan>3</colSpan>
        <columnIndex>0</columnIndex>
        <dashboardComponent>
            <autoselectColumnsFromReport>false</autoselectColumnsFromReport>
            <chartAxisRange>Auto</chartAxisRange>
            <chartSummary>
                <axisBinding>y</axisBinding>
                <column>RowCount</column>
            </chartSummary>
            <componentType>Column</componentType>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <groupingColumn>TITLE</groupingColumn>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/lead_rpt</report>
            <showPercentage>false</showPercentage>
            <showValues>false</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </dashboardComponent>
        <rowIndex>9</rowIndex>
        <rowSpan>3</rowSpan>
    </dashboardGridComponents>
    <numberOfColumns>9</numberOfColumns>
    <rowHeight>90</rowHeight>
</gridLayout>
<isGridLayout>true</isGridLayout>
<runningUser>admin@s1.com</runningUser>
<textColor>#000000</textColor>
<title>sfx</title>
<titleColor>#000000</titleColor>
```

```
    <titleSize>12</titleSize>
</Dashboard>
```

SEE ALSO:

Folder

Report

DataCategoryGroup

Represents a data category group. This type extends the Metadata metadata type and inherits its `fullName` field.

Warning: Using Metadata API to deploy category changes from one organization to another permanently removes categories and record categorizations that are not specified in your XML file. Salesforce recommends that you manually create data categories and record associations in an organization from Setup by entering *`Data Categories`* in the `Quick Find` box, then selecting **Data Categories** rather than deploying changes from a sandbox to a production organization. For more information, see Usage.

Data category groups are provided to:

- Classify and filter data.
- Share data among users.

Every data category group contains items or data categories that can be organized hierarchically.

The example below shows the `Geography` data category group and its data categories.

```
Geography
    Worldwide
        North America
            United States of America
            Canada
            Mexico
        Europe
        Asia
```

Note: See "What Are Data Categories?" in the Salesforce online help for more information on data category groups, data categories, parent and sub categories.

File Suffix and Directory Location

The file suffix is `.datacategorygroup`. There is one file for each data category group stored in the `datacategorygroups` folder in the corresponding package directory.

Version

Data category groups are available in API version 18.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`active`	boolean	Required. The status of the category group. Indicates whether this category group is active, (`true`), or not active (`false`).
`dataCategory`	DataCategory	Required. The top-level category within the data category group.
`description`	string	The description of the data category group.
`fullName`	string	Required. The unique name of the data category group. When creating a data category group, the `fullName` field and the file name (without its suffix) must match.The `fullName` 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.
`label`	string	Required. Label that represents the object in Salesforce.
`objectUsage`	ObjectUsage	The objects that are associated with the data category group.

DataCategory

Represents an item (or data category) in the data category group. A data category can recursively contain a list of other data categories.

Field Name	Field Type	Description
`dataCategory`	DataCategory[]	A recursive list of sub data categories. For example, a list of countries within a continent. You can create up to 100 categories in a data category group and have up to 5 levels in a data category group hierarchy.
`label`	string	Required. Label for the data category throughout the Salesforce user interface.
`name`	string	Required. The developer name of the data category used as a unique identifier for API access. 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. **Important:** The value for this field is defined once and cannot be changed later. **Warning:** If you deploy a category group that already exists in an organization, any category that is not defined in the XML file is permanently removed from your organization. For more information see Usage.

ObjectUsage

Represents the objects that can be associated with the data category group. This association allows the object to be classified and filtered using the data categories.

Field Name	Field Type	Description
`object`	string[]	A list of the object names that can be associated with the data category group. Valid values are: • `KnowledgeArticleVersion`—to associate articles. See "Modifying Category Group Assignments in Salesforce Knowledge" in the Salesforce online help for more information on data category groups association to articles. • `Question`—to associate questions. You can associate the `Question` object with at most one category group. See "Assigning Data Categories to Answers" in the Salesforce online help for more information on data category groups association to questions. Warning: If you deploy a category group that already exists in an organization, any object association that is not defined in the XML file is permanently removed from your organization. Ensure that your XML file specifies all the records associated with your category group in the organization. For more information see Usage.

Declarative Metadata Sample Definition

This sample is the definition of the `Geography` data category group and its data categories:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataCategoryGroup xmlns="http://soap.sforce.com/2006/04/metadata">
   <label>Geography</label>
   <description>Geography structure of service center locations</description>
   <fullName>geo</fullName>

   <dataCategory> <name>WW</name> <label>Worldwide</label>
      <dataCategory> <name>AMER</name> <label>North America</label>
         <dataCategory>
             <name>USA</name>
             <label>United States of America</label>
         </dataCategory>
         <dataCategory>
             <name>CAN</name>
             <label>Canada</label>
         </dataCategory>
         <dataCategory>
             <name>MEX</name>
             <label>Mexico</label>
         </dataCategory>
      </dataCategory>
      <dataCategory> <name>EMEA</name> <label>Europe, Middle East, Africa</label>
```

```
        <dataCategory>
            <name>FR</name>
            <label>France</label>
        </dataCategory>
        <dataCategory>
            <name>SP</name>
            <label>Spain</label>
       </dataCategory>
        <dataCategory>
           <name>UK</name>
           <label>United-Kingdom</label>
        </dataCategory>
     </dataCategory>
     <dataCategory>
         <name>APAC</name>
         <label>Asia</label>
     </dataCategory>
  </dataCategory>

  <objectUsage>
     <object>KnowledgeArticleVersion </object>
  <objectUsage>
</DataCategoryGroup>
```

Usage

When you deploy a category group XML file, Metadata API checks whether the category group exists in the target organization. If the category group does not exist, it is created. If the category group already exists, then Metadata API:

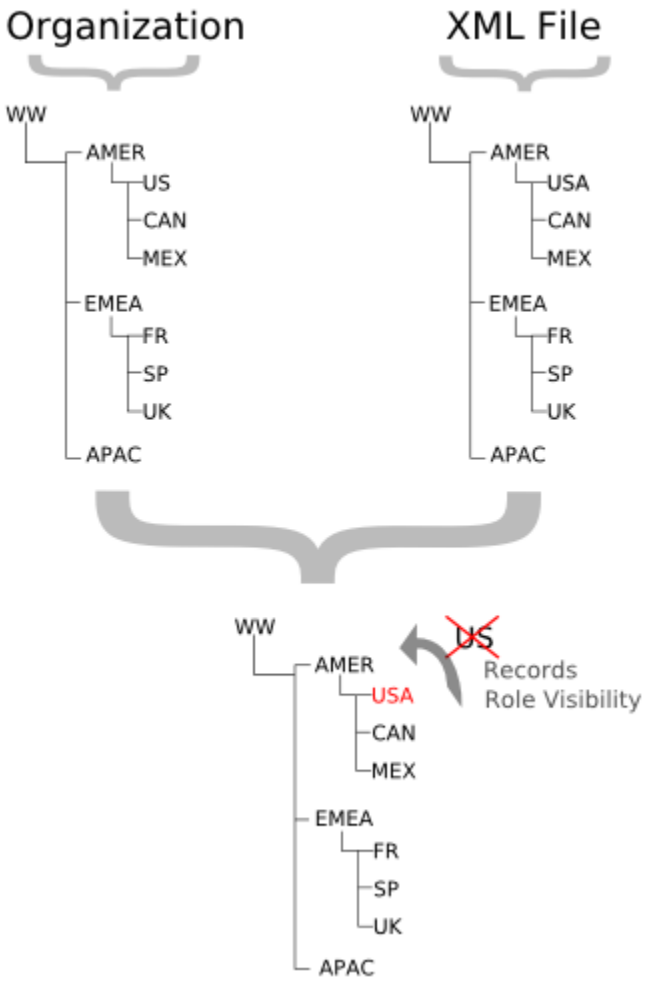
- Adds any new category or object defined in the XML file.
- Deletes any category that is not defined in the XML file. Records associated with the deleted categories are re-associated with the parent category.
- Deletes any object association that is not defined in the XML file.
- Moves any category if its hierarchical position differs from the position specified in the XML file.

  Note: When a category moves to a new parent category, users that have no visibility on the new parent category lose their visibility to the repositioned category.

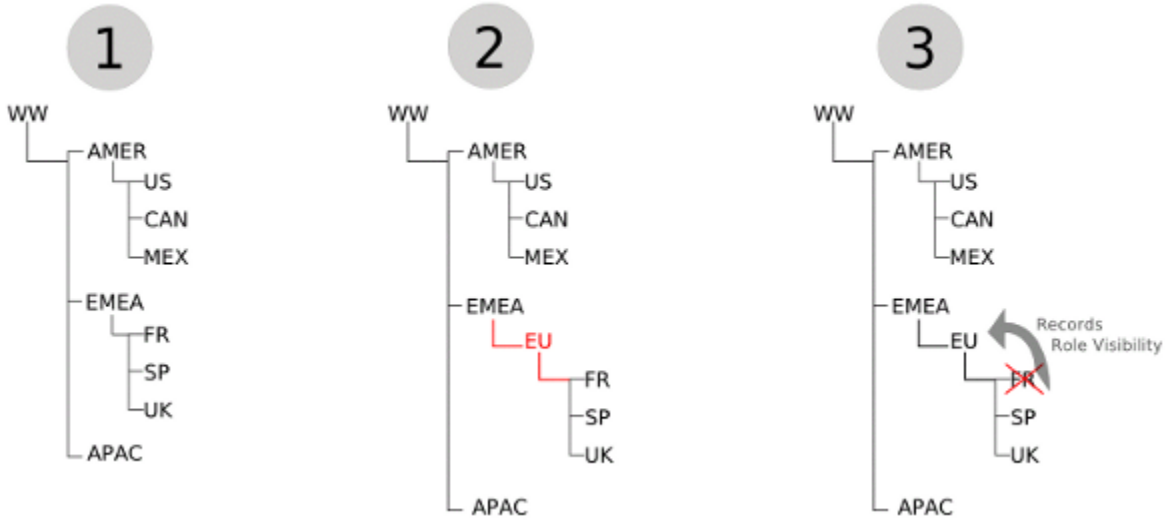
Note: For more information about category deletion, category repositioning and its impact on record categorization and visibility see "Deleting Data Categories" and "Modifying and Positioning Data Categories" in the Salesforce online help.

Using Metadata API to deploy category changes from one organization to another permanently removes categories and record categorizations that are not specified in your XML file. Salesforce recommends that you manually create data categories and record associations in an organization from Setup by entering *Data Categories* in the Quick Find box, then selecting **Data Categories** rather than deploying changes from a sandbox to a production organization.

The following example illustrates what happens if you deploy an XML representation of a Geography data category group hierarchy to an organization that already has this data category group defined. Note that the organization contains a US category, while the XML file includes a USA category in the same hierarchical position. The Metadata API deployment process deletes the US category from the organization and moves associations for any records from US to the parent AMER category. It also adds the USA category under AMER. Note that all records that were previously categorized with US are now associated with the AMER category.



The next example illustrates what can happen when you delete or move a category in a data category group and deploy its XML representation from a sandbox to a production organization that already has this data category group defined. Hierarchy 1 shows the initial data category group in the sandbox organization. In hierarchy 2, we add an `EU` category under `EMEA` and move `FR`, `SP` and `UK` below `EU`. In hierarchy 3, we delete `FR` and associate its records with its new parent, `EU`. Finally, we deploy the changes from the sandbox to the production organization.



Metadata API has no concept of the order of the changes made to the sandbox organization. It just deploys the changes from one organization to another. During the deployment, it first notices the deletion of the `FR` category and removes it from the production organization. Consequently, it moves associations for any records from `FR` to its parent on the production organization, `EMEA`. Metadata API then adds the `EU` category and moves `SP` and `UK` below it. Although the category group hierarchy looks the same in both organizations, record categorization in production is different from the sandbox organization. The records that were originally associated with `FR` in hierarchy 1 are associated with `EU` in the sandbox organization, but are associated with `EMEA` in the production organization.

DelegateGroup

Represents a group of users who have the same administrative privileges. These groups are different from public groups used for sharing. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

DelegateGroup components have the suffix `.delegateGroup` and are stored in the `delegateGroups` folder. The file prefix must match the developer name of the delegate group. For example, a delegate group with a developer name of MyDelegateGroup would have a file name of `MyDelegateGroup.delegateGroup`.

Version

DelegateGroup components are available in API version 36.0 and later.

Special Access Rules

Only users with the "View Setup and Configuration" permission can be delegated administrators.

Fields

Field Name	Field Type	Description
customObjects	string[]	The custom objects associated with the group. Delegated administrators can customize nearly every aspect of each of those custom objects, including creating a custom tab. However, they cannot create or modify relationships on the objects or set organization-wide sharing defaults. Delegated administrators must have access to custom objects to access the merge fields on those objects from formulas.
groups	string[]	The groups with users assigned by delegated administrators.
label	string	Required. The delegated group's non-API name.
loginAccess	boolean	Required. Allows users in this group to log in as users in the role hierarchy that they administer (`true`) or not (`false`). Depending on your organization settings, individual users must grant login access to allow their administrators to log in as them.
permissionSets	string[]	The permission sets assignable to users in specified roles and all subordinate roles by delegated administrators.
profiles	string[]	The profiles assignable to users by delegated administrators.
roles	string[]	The roles and subordinates for which delegated administrators of the group can create and edit users.

Declarative Metadata Sample Definition

The following is an example of a DelegateGroup component.

```
<?xml version="1.0" encoding="UTF-8"?>
<DelegateGroup xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>MyDelegateGroup</label>
    <loginAccess>true</loginAccess>
    <name>MyDelegateGroup</name>
    <profiles>Chatter Free User</profiles>
    <profiles>Chatter Moderator User</profiles>
    <profiles>Marketing User</profiles>
    <permissionSets>My Permset</permissionSets>
    <roles>LesserBossMan</roles>
</DelegateGroup>
```

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>*</members>
        <name>DelegateGroup</name>
    </types>
    <version>36.0</version>
</Package>
```

Document

Represents a Document. All documents must be in a document folder, for example `sampleFolder/TestDocument`. This metadata type extends the MetadataWithContent component and shares its fields.

Currently, users are not able to export document metadata to a local file system using the Force.com IDE.

Retrieving Documents

You can't use the wildcard (*) symbol with documents in `package.xml`. To retrieve the list of documents for populating `package.xml` with explicit names, call `listMetadata()` and pass in `DocumentFolder` as the type. Note that DocumentFolder is not returned as a type in `describeMetadata()`. Document is returned from `describeMetadata()` with an associated attribute of `inFolder` set to true. If that attribute is set to true, you can construct the type by using the component name with the word Folder, such as DocumentFolder.

The following example shows folders in `package.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
        <name>Report</name>
    </types>
    <version>36.0</version>
</Package>
```

For each document an accompanying metadata file named *DocumentFilename*`-meta.xml` is created in the document folder. For example, for a document `TestDocument.png` in the sampleFolder folder, there's a `TestDocument.png-meta.xml` in the `documents/sampleFolder` of the package.

Version

Documents are available in API version 10.0 and later.

In API version 17.0 and later, you can delete a folder containing documents moved to the Recycle Bin. When you delete the folder, any related documents in the Recycle Bin are permanently deleted.

In API version 18.0 and later, documents do not need an extension.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`content`	base64	Content of the document. 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.
`description`	string	A description of the document. Enter a description to distinguish this document from others.
`fullName`	string	The name of the document, including the folder name. In version 17.0 and earlier, the `fullName` included the document extension. In version 18.0 and later, the `fullName` does not include the file extension. The `fullName` 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. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the `name` field. This field is inherited from the Metadata component.
`internalUseOnly`	boolean	Required. Indicates whether the document is confidential (`true`) or not (`false`). This field and `public` are mutually exclusive; you cannot set both to `true`.
`keywords`	string	Contains one or more words that describe the document. A check for matches to words in this field is performed when doing a search.
`name`	string	The list of characters allowed in the `fullName` field has been reduced for versions 14.0 and later. This field contains the value contained in the `fullName` field before version 14.0. This field is only populated if the value of the `fullName` field contained characters that are no longer accepted in that field.
`public`	boolean	Required. Indicates whether the document is an image available for HTML email templates and does not require a Salesforce username and password to view in an email (`true`) or not (`false`). If the images will be used as a custom app logo or custom tab icon, both of which require a Salesforce username and password to view, set this field to `false`. This field and `internalUseOnly` are mutually exclusive; you cannot set both to `true`.

Declarative Metadata Sample Definition

The following is the definition of a document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="http://soap.sforce.com/2006/04/metadata">
    <internalUseOnly>false</internalUseOnly>
    <name>Q2 Campaign Analysis</name>
    <public>false</public>
    <description>Analyze Q2 campaign effectiveness</description>
</Document>
```

SEE ALSO:

Folder

EmailTemplate

Represents an email template. This metadata type extends the MetadataWithContent component and shares its fields.

File Suffix and Directory Location

The file suffix is `.email` for the template file. The accompanying metadata file is named *`EmailTemplateName`*`-meta.xml`.

EmailTemplate components are stored in the `email` folder in the corresponding package directory. For example, for an email template named SampleTemplate in the sampleFolder folder, there's a `SampleTemplate-meta.xml` in the `email/sampleFolder` of the package.

Retrieving Email Templates

You can't use the wildcard (*) symbol with email templates in `package.xml`. To retrieve the list of email templates for populating `package.xml` with explicit names, call `listMetadata()` and pass in `EmailTemplate` as the type.

The following example shows folders in `package.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
```

```
        <name>Report</name>
    </types>
    <version>36.0</version>
</Package>
```

Version

Email templates are available in API version 12.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`apiVersion`	double	The API version if this is a Visualforce email template. Every Visualforce email template has an API version specified at creation. This field is available in API version 16.0 and later.
`attachedDocuments`	string[]	A list of references to documents in your organization. These documents are included as attachments in the email template. Each document is referenced by its path, for example `MyFolder/MyDocument.txt`.
`attachments`	Attachment[]	A list of attachments for the email template.
`available`	boolean	Required. Indicates whether this template is offered to users when sending an email (`true`) or not (`false`).
`content`	base64Binary	Content of the email template. 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 contains: • Binary content of the email body if `type` is set to `text` • HTML email content if `type` is set to `html` • HTML body if `type` is set to `custom` • Visualforce body if `type` is set to `visualforce` This field is inherited from the MetadataWithContent component.
`description`	string	The email template description. This can be useful to describe the reason for creating the template.
`encodingKey`	Encoding (enumeration of type string)	Required. The default encoding setting is Unicode: `UTF-8`. Change it if your template requires data in a different format. Valid values include: • `UTF-8`—Unicode (UTF-8) • `ISO-8859-1`—General US & Western Europe (ISO-8859–1, ISO-LATIN-1) • `Shift_JIS`—Japanese (Shift-JIS)

Field Name	Field Type	Description
		• `ISO-2022-JP`—Japanese (JIS) • `EUC-JP`—Japanese (EUC-JP) • `x-SJIS_0213`—Japanese (Shift-JIS_2004) • `ks_c_5601-1987`—Korean (ks_c_5601-1987) • `Big5`—Traditional Chinese (Big5) • `GB2312`—Simplified Chinese (GB2312) • `Big5-HKSCS`—Traditional Chinese Hong Kong (Big5–HKSCS)
`fullName`	string	The email template developer name used as a 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 end with an underscore, and not contain two consecutive underscores. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the `name` field. This field is inherited from the Metadata component.
`letterhead`	string	The letterhead name associated with this email template. Only available when `type` is set to `html`.
`name`	string	Required. Email template name. The list of characters allowed in the `fullName` field has been reduced for versions 14.0 and later. This field contains the value contained in the `fullName` field before version 14.0.
`packageVersions`	PackageVersion[]	The list of package versions for any managed packages containing components that are referenced by this email template. This field is only relevant for Visualforce email templates. For more information about managed packages, see the Force.com Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.
`style`	EmailTemplateStyle (enumeration of type string)	Required. The style of the template. This field is only available when `type` is set to `html`. Valid style values include: • `none` • `freeForm` • `formalLetter` • `promotionRight` • `promotionLeft` • `newsletter` • `products`
`subject`	string	The email subject.
`textOnly`	string	The text of the email body if `type` is set to `html` or `custom`.

Field Name	Field Type	Description
`type`	EmailTemplateType (enumeration of type string)	Required. The email template type. The valid values are: • `text` -all users can create or change text email templates. • `html` - administrators and users with the "Edit HTML Templates" permission can create HTML email templates based on a letterhead. • `custom` - administrators and users with the "Edit HTML Templates" permission can create custom HTML email templates without using a letterhead. You must either know HTML or obtain the HTML code to insert in your email template. • `visualforce` - administrators and users with the "Customize Application" permission can create email templates using Visualforce.

Attachment

Attachment represents an email attachment.

Field	Field Type	Description
`content`	base64Binary	Required. The attachment content. 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.
`name`	string	Required. The attachment file name.

Declarative Metadata Sample Definition

A sample XML definition of an email template is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmailTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
    <available>true</available>
    <description>Sample Email Template</description>
    <encodingKey>ISO-8859-1</encodingKey>
    <name>Sample Email Template</name>
    <style>none</style>
    <subject>Sample email subject</subject>
    <textOnly>Your case has been resolved.</textOnly>
    <type>custom</type>
</EmailTemplate>
```

SEE ALSO:

Letterhead

EntitlementProcess

Represents the settings for an entitlement process. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Entitlement process values are stored in files in the `entitlementProcesses` directory. Each file has the name of a process and the suffix `.entitlementProcess`. Each file contains one entitlement process or, if entitlement versioning is enabled, one version of an entitlement process. The name of the file is the name of the entitlement process with the version appended to the end, if applicable (for example, an entitlement process named "gold_support" might have the file name "gold_support_v2.entitlementProcess"). This file name corresponds to the `slaProcess.NameNorm` field exposed through the SOAP API. This file name is distinct from the `name` field, which represents what displays in the user interface and, if versioning is enabled, might be shared among multiple versions of the same entitlement process.

Version

Entitlement processes are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Indicates whether the entitlement process is active (`true`) or not (`false`).
`businessHours`	string	The business hours that apply to the entitlement process. This field is available in API version 30.0 and later.
`description`	string	The description of the entitlement process.
`entryStartDateField`	string	For milestone processes on which a case enters the process based on a custom date/time field on the case, specifies which date and time are used. Valid values are: • SlaStartDate (entitlement process start date) • CreatedDate (date case was opened) • ClosedDate (date case was closed) • LastModifiedDate (date case was last modified) • StopStartDate (date case was stopped)
`exitCriteriaBooleanFilter`	string	For milestone processes on which a case exits the process when custom criteria are met, and for which filter logic is added, specifies that logic.
`exitCriteriaFilterItems`	FilterItem[]	For milestone processes on which a case exits the process when custom criteria are met, specifies those criteria.

Field Name	Field Type	Description
exitCriteriaFormula	string	For milestone processes on which a case exits the process when a custom formula evaluates to true, specifies that formula.
isVersionDefault	boolean	Indicates whether the entitlement process is the default version (true) or not (false). This field is available in API version 28.0 and later.
milestones	EntitlementProcessMilestoneItem[]	Represents a milestone on the entitlement process.
name	string	The name of the entitlement process as it displays in the user interface.
versionMaster	string	Identifies the sequence of versions to which this entitlement process belongs. This field's contents can be any value as long as it is identical among all versions of the entitlement process. This field is available in API version 28.0 and later.
versionNotes	string	The description of the entitlement process version. This field is available in API version 28.0 and later.
versionNumber	int	The version number of the entitlement process. Must be 1 or greater. This field is available in API version 28.0 and later.

EntitlementProcessMilestoneItem

Represents a milestone item on an entitlement process.

Fields

Field Name	Field Type	Description
businessHours	string	The business hours that apply to the milestone. This field is available in API version 30.0 and later.
criteriaBooleanFilter	string	For milestones that apply only when criteria are met and for which filter logic is added, specifies that logic.
milestoneCriteriaFilterItems	FilterItem[]	For milestones that apply only when criteria are met, specifies those criteria.
milestoneCriteriaFormula	string	For milestones that apply only when a formula evaluates to true, specifies that formula.
milestoneName	string	The name of the milestone.

Field Name	Field Type	Description
minutesCustomClass	string	The name of the Apex class that is used to calculate the trigger time. This field is available in API version 30.0 and later.
minutesToComplete	int	The number of minutes from when the case enters the entitlement process that the milestone occurs.
successActions	WorkflowActionReference[]	The actions triggered when the milestone is completed.
timeTriggers	EntitlementProcessMilestoneTimeTrigger[]	The time triggers on an entitlement process milestone.
useCriteriaStartTime	boolean	When the milestone starts: when the milestone criteria are met (true) or when the case enters the entitlement process (false).

EntitlementProcessMilestoneTimeTrigger

Represents the time trigger on an entitlement process milestone.

Fields

Field Name	Field Type	Description
actions	WorkflowActionReference[]	The actions to take when the time trigger is reached, if, at that time, the milestone is not completed.
timeLength	int	The length of time between the time trigger activation and the milestone target completion date. This may be a negative or positive value. Negative values indicate that the target completion date has not yet arrived and correspond to warning time triggers. Positive values indicate that the target completion date has passed and correspond to violation time triggers.
workflowTimeTriggerUnit	MilestoneTimeUnits (enumeration of type string)	Specifies the type of unit used to determine when a workflow should be triggered. Valid values are: • Minutes • Hours • Days

Declarative Metadata Sample Definition

This is a sample entitlement process.

```
<?xml version="1.0" encoding="UTF-8"?>
<EntitlementProcess xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <description>eppersone</description>
```

```
    <entryStartDateField>SlaStartDate</entryStartDateField>
    <exitCriteriaBooleanFilter>1 OR 2</exitCriteriaBooleanFilter>
    <exitCriteriaFilterItems>
        <field>Case.IsClosed</field>
        <operation>equals</operation>
        <value>true</value>
    </exitCriteriaFilterItems>
    <exitCriteriaFilterItems>
        <field>Case.Description</field>
        <operation>startsWith</operation>
        <value>foo</value>
    </exitCriteriaFilterItems>
    <milestones>
        <milestoneName>m1</milestoneName>
        <minutesToComplete>1</minutesToComplete>
        <successActions>
            <name>emailBob</name>
            <type>Alert</type>
        </successActions>
        <timeTriggers>
            <actions>
                <name>emailAlice</name>
                <type>Alert</type>
            </actions>
            <actions>
                <name>setEscalateToTrue</name>
                <type>FieldUpdate</type>
            </actions>
            <timeLength>1</timeLength>
            <workflowTimeTriggerUnit>Minutes</workflowTimeTriggerUnit>
        </timeTriggers>
        <timeTriggers>
            <actions>
                <name>setStopToTrue</name>
                <type>FieldUpdate</type>
            </actions>
            <timeLength>2</timeLength>
            <workflowTimeTriggerUnit>Minutes</workflowTimeTriggerUnit>
        </timeTriggers>
        <useCriteriaStartTime>false</useCriteriaStartTime>
    </milestones>
    <milestones>
        <milestoneCriteriaFilterItems>
            <field>Case.Priority</field>
            <operation>equals</operation>
            <value>High</value>
        </milestoneCriteriaFilterItems>
        <milestoneName>m2</milestoneName>
        <minutesToComplete>120</minutesToComplete>
        <useCriteriaStartTime>true</useCriteriaStartTime>
        <successActions>
            <name>emailBob</name>
            <type>Alert</type>
        </successActions>
```

```
    </milestones>
</EntitlementProcess>
```

EntitlementTemplate

Represents an entitlement template. Entitlement templates are predefined terms of customer support that you can quickly add to products. For example, you can create entitlement templates for Web or phone support so that users can easily add entitlements to products offered to customers. EntitlementTemplate extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

EntitlementTemplate components are stored in the `entitlementTemplates` directory of the corresponding package directory. The file name matches the unique name of the entitlement template, and the extension is `.entitlementTemplate`.

Version

Force.com EntitlementTemplate components are available in API version 18.0 and higher.

Fields

Field	Field Type	Description
`businessHours`	string	The entitlement's supported business hours.
`casesPerEntitlement`	int	Lets you limit the number of cases the entitlement supports.
`entitlementProcess`	string	The entitlement process associated with the entitlement.
`isPerIncident`	boolean	`true` if entitlements created from this template service a limited number of cases; `false` otherwise.
`term`	int	The number of days the entitlement is in effect.
`type`	string	The type of entitlement, such as Web or phone support.

Declarative Metadata Sample Definition

A sample XML definition of an entitlement template is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<EntitlementTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
    <businessHours>AlternateBusinessHours</businessHours>
    <casesPerEntitlement>12</casesPerEntitlement>
    <entitlementProcess>Process1</entitlementProcess>
    <isPerIncident>true</isPerIncident>
    <term>33</term>
    <type>Phone Support</type>
</EntitlementTemplate>
```

EscalationRules

Represents case escalation rules to escalate cases automatically if they are not resolved within a certain period of time. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object. The `package.xml` syntax for accessing all escalation rules for all objects is:

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

All rules for a specific object uses a similar syntax without the wildcard. For example, all escalation rules for the Case object would use this syntax:

```
<types>
    <members>Case</members>
    <name>EscalationRules</name>
</types>
```

You can also access specific escalation rules for an object. The following example only accesses the "samplerule" and "newrule" escalation rules on the Case object. Notice that for this example the type name syntax is `EscalationRule` and not `EscalationRules`.

```
<types>
    <members>Case.samplerule</members>
    <members>Case.newrule</members>
    <name>EscalationRule</name>
</types>
```

File Suffix and Directory Location

EscalationRules for an object have the suffix `.escalationRules` and are stored in the `escalationRules` folder. For example, all Case escalation rules are stored in the `Case.escalationRules` file.

Version

EscalationRules components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`escalationRule`	EscalationRule[]	Represents one escalation rule and specifies whether it is active or not. Escalation rules are processed in the order they appear in the EscalationRules container.

EscalationRule

Field Name	Field Type	Description
`active`	boolean	Indicates whether the escalation rule is active (`true`) or not (`false`).
`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 `create()` to see an example of this field specified for a call. This value cannot be `null`.
`ruleEntry`	`RuleEntry[]`	Contains the definitions of the rule entries in the escalation rule.

RuleEntry

Represents the fields used by the rule.

Field Name	Field Type	Description
`booleanFilter`	string	Advanced filter conditions that were specified for the rule.
`businessHours`	string	The hours at which escalation actions are performed. Specify only if `businessHoursSource` is set to `Static`.
`businessHoursSource`	`BusinessHoursSourceType` (enumeration of type string)	Valid values are: • `None` • `Case` • `Static`
`criteriaItems`	`FilterItem`	The items in the list that define the assignment criteria.
`disableEscalationWhenModified`	boolean	Indicates whether the escalation is disabled when the record is modified `true`) or not (`false`).
`escalationAction`	`EscalationAction[]` on page 325	The actions to perform when the escalation criteria are met.
`escalationStartTime`	`EscalationStartTimeType` (enumeration of type string)	Indicates the start time for the escalation. Valid values are: • `CaseCreation` • `CaseLastModified`
`formula`	string	The validation formula. Note: Specify either `formula` or `criteriaItems`, but not both fields.

EscalationAction

Describes the action to take for an escalation rule.

Field Name	Field Type	Description
assignedTo	string	The name of the user or queue the item is assigned to.
assignedToTemplate	string	Specifies the template to use for the email that is automatically sent to the new owner specified by the escalation rule.
assignedToType	AssignToLookupValueType (enumeration of type string)	Valid values are: • User • Queue
minutesToEscalation	int	The number of minutes until the escalation occurs.
notifyCaseOwner	boolean	Indicates that the owner of the case is notified when the case is escalated (true) or not (false).
notifyEmail	string	Specifies the email address of the user to notify.
notifyTo	string	Specifies the user to notify.
notifyToTemplate	string	Specifies the template to user for the notification email.

Declarative Metadata Sample Definition

The following is an example EscalationRules component:

```
<EscalationRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <escalationRule>
        <fullName>samplerule</fullName>
        <active>false</active>
        <ruleEntry>
            <businessHours>test</businessHours>
            <businessHoursSource>Static</businessHoursSource>
            <criteriaItems>
                <field>Case.Description</field>
                <operation>contains</operation>
                <value>test</value>
            </criteriaItems>
            <escalationAction>
                <assignedTo>someuser@org.com</assignedTo>
                <assignedToTemplate>emailtemplatename</assignedToTemplate>
                <assignedToType>User</assignedToType>
                <minutesToEscalation>1440</minutesToEscalation>
                <notifyCaseOwner>false</notifyCaseOwner>
            </escalationAction>
            <escalationStartTime>CaseLastModified</escalationStartTime>
        </ruleEntry>
```

```
    </escalationRule>
</EscalationRules>
```

ExternalDataSource

Represents the metadata associated with an external data source. Create external data sources to manage connection details for integration with data and content that are stored outside your Salesforce org.

This type 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 suffix `.dataSource`, and the prefix 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
`authProvider`	string	The authentication provider that is represented by the AuthProvider component.
`certificate`	string	If you specify a certificate, your Salesforce org supplies it when establishing each two-way SSL connection with the external system. The certificate is used for digital signatures, which verify that requests are coming from your Salesforce org.
`customConfiguration`	string	A string of configuration parameters that are specific to the external data source's `type`. • customConfiguration for Lightning Connect—Salesforce Adapter • customConfiguration for Lightning Connect—OData 2.0 or 4.0 Adapter • customConfiguration for Lightning Connect—Custom Adapter
`endpoint`	string	The URL of the external system, or if that URL is defined in a named credential, the named credential URL. A named credential URL contains the scheme *`callout:`*, the name of the named credential, and an optional path. For example: callout:*`My_Named_Credential/some_path`*. Corresponds to `URL` in the user interface.
`isWritable`	boolean	Allows users to create, update, or delete external object records for this data source from within Salesforce. By default, external objects are

Field Name	Field Type	Description
		read-only. Corresponds to `Allow Create, Edit, and Delete` in the user interface. Writable external objects aren't supported with the Salesforce adapter for Lightning Connect.
`label`	string	A user-friendly name for the external data source. The label is displayed in the Salesforce user interface, such as in list views. Examples include Acme Team Marketing Site, or Acme SharePoint.
`oauthRefreshToken`	string	The OAuth refresh token. Used to obtain a new access token for an end user when a token expires.
`oauthScope`	string	Specifies the scope of permissions to request for the access token. Corresponds to `Scope` in the user interface.
`oauthToken`	string	The access token issued by the external system.
`password`	string	The password to be used by your org to access the external system. Ensure that the credentials you use have adequate privileges to access the external system, perform searches, return data, and return information about the external system's metadata.
`principalType`	`ExternalPrincipalType` (enumeration of type string)	Determines whether you're using one set or multiple sets of credentials to access the external system. Corresponds to `Identity Type` in the user interface. The valid values are: • `Anonymous` • `PerUser` • `NamedUser`
`protocol`	`AuthenticationProtocol` (enumeration of type string)	The authentication protocol that's required to access the external system. The valid values are: • `NoAuthentication` • `Oauth` • `Password` For cloud-based Files Connect external systems, select **Oauth 2.0**. For on-premises systems, select **Password Authentication**. For Simple URL data sources, select **No Authentication**.
`repository`	string	Used for SharePoint Online. If metadata is not accessible, use this field to create tables and default table fields.
`type`	`ExternalDataSourceType` (enumeration of type string)	For Lightning Connect, specifies the adapter that connects to the external system. The valid values are: • `OData`—OData 2.0 adapter • `OData4`—OData 4.0 adapter • `SfdcOrg`—Salesforce adapter

Field Name	Field Type	Description
		• *ApexClassId*—DataSource.Provider class that defines the custom adapter created via the Apex Connector Framework For Files Connect, specifies the data source type. The valid values are: • ContentHubSharepoint—SharePoint 2010 or 2013 • ContentHubSharepointOffice365—SharePoint Online • ContentHubSharepointOneDrive—OneDrive for Business • ContentHubGDrive—Google Drive If Chatter is enabled, you can also specify SimpleURL to access data hosted on a web server that doesn't require authentication. The Identity and Wrapper types are reserved for future use.
username	string	The username to be used by your org to access the external system. Ensure that the credentials you use have adequate privileges to access the external system, perform searches, return data, and return information about the external system's metadata.
version	string	Reserved for future use.

customConfiguration for Lightning Connect—Salesforce Adapter

The following sample JSON-encoded configuration string defines parameters that apply when the external data source's type is set to SfdcOrg.

```
{"apiVersion":"32.0","environment":"CUSTOM",
"searchEnabled":"true","timeout":"120"}
```

The parameters correspond to these fields in the user interface:

- apiVersion—API Version
- environment—Connect to
- searchEnabled—Include in Salesforce Searches
- timeout—Connection Timeout

customConfiguration for Lightning Connect—OData 2.0 or 4.0 Adapter

The following JSON-encoded configuration string defines parameters that apply when the external data source's type is set to OData or OData4.

```
{"inlineCountEnabled":"true","csrfTokenName":"X-CSRF-Token",
"requestCompression":"false","pagination":"CLIENT",
"noIdMapping":"false","format":"ATOM",
"searchFunc":"","compatibility":"DEFAULT",
"csrfTokenEnabled":"true","timeout":"120",
"searchEnabled":"true"}
```

The parameters correspond to these fields in the user interface.

- compatibility—Special Compatibility
- csrfTokenEnabled—CSRF Protection
- csrfTokenName—Anti-CSRF Token Name
- format—Format
- inlineCountEnabled—Request Row Counts
- noIdMapping—High Data Volume
- pagination—Server Driven Pagination
- requestCompression—Compress Requests
- searchEnabled—Include in Salesforce Searches
- searchFunc—Custom Query Option for Salesforce Search
- timeout—Connection Timeout

`customConfiguration` for Lightning Connect—Custom Adapter

The following sample JSON-encoded configuration string defines the parameter that applies when the external data source's `type` is set to the ID of a `DataSource.Provider` class.

```
{"noIdMapping":"false"}
```

The `noIdMapping` parameter corresponds to the `High Data Volume` field in the user interface.

Declarative Metadata Sample Definition

The following is the definition of an external data source for Lightning Connect—OData 2.0 or 4.0 adapter.

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalDataSource xmlns="http://soap.sforce.com/2006/04/metadata">
    <authProvider>FacebookAuth</authProvider>
    <customConfiguration>{"compatibility":"DEFAULT",
    "noIdMapping":"false","inlineCountEnabled":"true",
    "searchEnabled":"true","format":"ATOM",
    "requestCompression":"false","pagination":"SERVER",
    "timeout":"120"}</customConfiguration>
    <endpoint>http://myappname.herokuapp.com/DataHub.svc</endpoint>
    <label>DataHub</label>
    <principalType>NamedUser</principalType>
    <protocol>Oauth</protocol>
    <type>OData</type>
</ExternalDataSource>
```

FlexiPage

Represents the metadata associated with a Lightning Page. A Lightning Page is the home page for an app that appears as a menu item in the Salesforce1 navigation menu. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: These app pages are known as FlexiPages in the API, but are referred to as Lightning Pages in the rest of the Salesforce documentation and UI.

For more information on Lightning Pages, see the Salesforce Help.

Note: The FlexiPage component is supported only in Salesforce1. For more information about Salesforce1, see the Salesforce Help.

File Suffix and Directory Location

FlexiPage components have the suffix `.flexipage` and are stored in the `flexipages` folder.

Version

FlexiPage components are available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`description`	string	The optional description text of the Lightning Page.
`flexiPageRegions`	FlexiPageRegion[]	Required. The list of regions of a page.
`masterLabel`	string	Required. The label for this FlexiPage, which displays in Setup.
`pageTemplate`	string	Required. The template associated with the FlexiPage. This field is available in API version 33.0 or later.
`parentFlexiPage`	string	This field is reserved for future use. This field is available in API version 35.0 or later.
`platformActionList`	PlatformActionList	The list of actions, and their order, that display in the Salesforce1 action bar for the Lightning Page. This field is available in API version 34.0 and later.
`quickActionList`	QuickActionList	The list of quick actions associated with the Lightning Page.
`sobjectType`	string	This field is reserved for future use. This field is available in API version 33.0 or later.
`type`	FlexiPageType (enumeration of type string)	Required. The type of a page. In API version 32.0, this field can only have a value of `AppPage`. Valid values are: • `AppPage`—A Lightning Page used as the home page for a custom app. This field is available in API version 32.0 and later.

FlexiPageRegion

FlexiPage Region represents the properties of a region of a page. A region can contain a record list component or a recent items component that can be scoped to a set of entities.

Field Name	Field Type	Description
`appendable`	RegionFlagStatus (enumeration of type string)	This field is reserved for future use. Valid values are: • `disabled` • `enabled` This field is available in API version 35.0 or later.
`componentInstances`	ComponentInstance[]	Properties and name of the component instance.
`mode`	FlexiPageRegionMode (enumeration of type string)	This field is reserved for future use. Valid values are: • `Append` • `Prepend` • `Replace` This field is available in API version 35.0 or later.
`name`	string	Required. Unique name of the FlexiPage region.
`prependable`	RegionFlagStatus (enumeration of type string)	This field is reserved for future use. Valid values are: • `disabled` • `enabled` This field is available in API version 35.0 or later.
`replaceable`	RegionFlagStatus (enumeration of type string)	This field is reserved for future use. Valid values are: • `disabled` • `enabled` This field is available in API version 35.0 or later.
`type`	FlexiPageRegionType (enumeration of type string)	Required. The type of FlexiPage region. Valid values are: • `Facet` • `Region` This field is available in API version 35.0 or later.

ComponentInstance

Instance of a component in a page, such as a filter list.

Field Name	Field Type	Description
`componentInstanceProperties`	ComponentInstanceProperty[]	The value of a single property in a component instance. A component instance can have no properties.
`componentName`	string	Required. The name of a single instance of a component.

ComponentInstanceProperty

Value of a single property in a component instance.

Field Name	Field Type	Description
`name`	string	Name of the property, unique within the component instance.
`value`	string	Reference or value of the property.

When you give a standard label to a tab in a Tabs component—such as Activity, Collaborate, or Details—and when the `name` field is set to `title`, the `value` field uses a system-defined value instead of the label.

- `Standard.Tab.activity`
- `Standard.Tab.collaborate`
- `Standard.Tab.detail`
- `Standard.Tab.feed`
- `Standard.Tab.preview`
- `Standard.Tab.relatedLists`

For example, let's say you have a Lightning Page that contains a tab with the standard label "Activity". If you query the definition that page, you see the system-defined name of the tab, not the label, in `value`.

```
<componentInstances>
  <componentInstanceProperties>
    <name>title</name>
    <value>Standard.Tab.activity</value>
  </componentInstanceProperties>
    <componentName>flexipage:tab</componentName>
</componentInstances>
```

PlatformActionList

PlatformActionList represents the list of actions, and their order, that display in the Salesforce1 action bar for the layout. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionListContext`	PlatformActionListContext (enumeration of type string)	Required. The context of the action list. Valid values are: • `BannerPhoto` • `Chatter` • `Dockable` • `FeedElement` • `FlexiPage` • `Global` • `ListView` • `ListViewDefinition` • `ListViewRecord` • `Lookup` • `MruList` • `MruRow` • `ObjectHomeChart` • `Photo` • `Record` • `RecordEdit` • `RelatedList` • `RelatedListRecord`
`platformActionListItems`	PlatformActionListItem[]	The actions in the PlatformActionList.
`relatedSourceEntity`	string	When the `ActionListContext` is RelatedList or RelatedListRecord, this field represents the API name of the related list to which the action belongs.

PlatformActionListItem

PlatformActionListItem represents an action in the PlatformActionList. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionName`	string	The API name for the action in the list.
`actionType`	PlatformActionType (enumeration of type string)	The type of action. Valid values are: • `ActionLink`—An indicator on a feed element that targets an API, a web page, or a file, represented by a button in the Salesforce Chatter feed UI. • `CustomButton`—When clicked, opens a URL or a Visualforce page in a window or executes JavaScript. • `InvocableAction`

Field Name	Field Type	Description
		• `ProductivityAction`—Productivity actions are predefined by Salesforce and are attached to a limited set of objects. You can't edit or delete productivity actions. • `QuickAction`—A global or object-specific action. • `StandardButton`—A predefined Salesforce button such as New, Edit, and Delete.
`sortOrder`	int	The placement of the action in the list.
`subtype`	string	The subtype of the action. For quick actions, the subtype is `QuickActionType`. For custom buttons, the subtype is `WebLinkTypeEnum`. For action links, subtypes are `Api`, `ApiAsync`, `Download`, and `Ui`. Standard buttons and productivity actions have no subtype.

Declarative Metadata Sample Definition

Here's a sample XML FlexiPage component definition for a travel app that tracks the user's trips, expense reports, and other relevant data:

```
<?xml version="1.0" encoding="UTF-8"?>
<FlexiPage xmlns="http://soap.sforce.com/2006/04/metadata">
   <flexiPageRegions>
   <description>Page to view recent trips</description>
      <componentInstances>
         <componentInstanceProperties>
            <name>entityName</name>
            <value>Trips__c</value>
         </componentInstanceProperties>
         <componentInstanceProperties>
            <name>filterName</name>
            <value>My_Trips</value>
         </componentInstanceProperties>
         <componentName>flexipage:filterListCard</componentName>
      </componentInstances>
      <componentInstances>
         <componentInstanceProperties>
            <name>entityName</name>
            <value>Expense_Report__c</value>
         </componentInstanceProperties>
         <componentInstanceProperties>
            <name>filterName</name>
            <value>My_Reports</value>
         </componentInstanceProperties>
         <componentName>flexipage:filterListCard</componentName>
      </componentInstances>
      <componentInstances>
         <componentInstanceProperties>
            <name>entityNames</name>
            <value>["User","Trips__c","Expense__c","Receipt__c"]</value>
```

```
            </componentInstanceProperties>
            <componentName>flexipage:recentItems</componentName>
        </componentInstances>
        <name>main</name>
    </flexiPageRegions>
    <masterLabel>My Travel, Inc.</masterLabel>
    <quickActionList>
        <quickActionListItems>
            <quickActionName>customAction1</quickActionName>
        </quickActionListItems>
        <quickActionListItems>
            <quickActionName>customAction2</quickActionName>
        </quickActionListItems>
    </quickActionList>
</FlexiPage>
```

And, here's the sample `package.xml` file that references the FlexiPage component definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Travel, Inc.</fullName>
    <types>
        <members>TravelIncFlexiPage</members>
        <name>CustomTab</name>
    </types>
    <types>
        <members>TravelIncFlexiPage</members>
        <name>FlexiPage</name>
    </types>
    <types>
        <members>TravelIncQuickActions</members>
        <name>QuickAction</name>
    </types>
    <version>29.0</version>
</Package>
```

Flow

Represents the metadata associated with a flow. With Flow, you can create an application that navigates users through a series of screens to query and update records in the database. You can also execute logic and provide branching capability based on user input to build dynamic applications. For information about the corresponding UI-based flow building tool, see "Cloud Flow Designer" in the Salesforce Help.

When using the file-based Metadata API to work with flows, consider that:

- You can't use Metadata API to access a flow installed from a managed package.
- Flow filenames shouldn't contain spaces, which can cause errors at deployment. Heading and trailing spaces are allowed, but are trimmed during deployment.
- You can't overwrite an active flow or one that was once active when deploying a flow using Metadata API.
- You can create a new version of a flow by giving the file a new version number and deploying it.

Declarative Metadata File Suffix and Directory Location

Flows are stored in the `Flow` directory of the corresponding package directory. The file name matches the flow's unique full name, and the extension is `.flow`.

Version

The flow Metadata API is available in API version 24.0 and later.

Flow

This metadata type represents a valid definition of a flow. This type extends the Metadata metadata type and inherits its `fullName` field.

Field Name	Field Type	Description
`actionCalls`	FlowActionCall[]	An array of nodes that define calls to actions. This field is available in API version 31.0 and later.
`apexPluginCalls`	FlowApexPluginCall[]	An array of nodes that define calls to Apex plug-ins.
`assignments`	FlowAssignment[]	An array of assignment nodes.
`choices`	FlowChoice[]	An array of static choice options.
`constants`	FlowConstant[]	An array of constants.
`decisions`	FlowDecision[]	An array of decision nodes.
`description`	string	Description of the flow.
`dynamicChoiceSets`	FlowDynamicChoiceSet[]	An array that constructs a set of choice options based on a database lookup.
`formulas`	FlowFormula[]	An array of formulas.
`fullName`	string	Required; inherited from the Metadata component. Name of the file in Metadata API. The `fullName` consists of two parts, separated by a hyphen: • Unique name for the flow that contains only underscores and alphanumeric characters. It must be unique across the organization, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. • Version number for the flow. For example, "sampleFlow-3" specifies version 3 of the flow whose unique name is sampleFlow.

Field Name	Field Type	Description
`interviewLabel`	string	Label for the interview. This label helps users and administrators differentiate interviews from the same flow. In the user interface, this label appears in the Paused Flow Interviews component on the user's Home tab and in the Paused and Waiting Interviews list on the flow management page.
`label`	string	Required. Label for the flow.
`loops`	FlowLoop[]	An array of nodes for iterating through collections. This field is available in API version 30.0 and later.
`processMetadataValues`	FlowMetadataValue[]	Metadata values for the flow. This field is available in API version 31.0 and later.
`processType`	FlowProcessType (enumeration of type string)	The type of the flow, as determined by the active version (or latest version, if there's no active version). Valid values are: • `AutoLaunchedFlow`—A flow that doesn't require user interaction • `Flow`—A flow that requires user interaction because it contains one or more screens, choices, or dynamic choices These values are reserved for future use. • `ActionPlan` • `JourneyBuilderIntegration` • `LoginFlow` • `Workflow` • `UserProvisioningFlow` Across versions, you can change the type from `Flow` to `AutoLaunchedFlow` or vice versa. This field is available in API version 31.0 and later.
`recordCreates`	FlowRecordCreate[]	An array of nodes for creating records in the database.
`recordDeletes`	FlowRecordDelete[]	An array of nodes for deleting records in the database.
`recordLookups`	FlowRecordLookup[]	An array of nodes for looking up records in the database.
`recordUpdates`	FlowRecordUpdate[]	An array of nodes for updating records in the database.
`screens`	FlowScreen[]	An array of screen nodes.
`startElementReference`	string	Specifies which node or element is the starting point in the flow.
`steps`	FlowStep[]	An array of step nodes.
`subflows`	FlowSubflow[]	An array of subflows. This field is available in API version 25.0 and later.
`textTemplates`	FlowTextTemplate[]	An array of text templates.

Field Name	Field Type	Description
variables	FlowVariable[]	An array of variable definitions.
waits	FlowWait[]	An array of wait nodes. This field is available in API version 32.0 and later.

FlowActionCall

Defines a call to an action from the flow. It extends FlowNode.

Available in API version 31.0 and later.

Field Name	Field Type	Description
actionName	string	Required. Name for the action. Must be unique across actions with the same actionType.
actionType	InvocableActionType (enumeration of type string)	Required. The action type. Valid values are: • apex—invokes an Apex method that has the @invocableMethod annotation • chatterPost—posts to Chatter • contentWorkspaceEnableFolders—enables folders in a library • emailAlert—sends an email by referencing a workflow email alert • emailSimple—sends an email by using flow resources • quickAction—invokes a QuickAction • submit—submits a record for approval These values are reserved for future use. • flow • thanks • metricRefresh
connector	FlowConnector	Specifies which node to execute after this action call.
faultConnector	FlowConnector	Specifies which node to execute if the action call results in an error.
inputParameters	FlowActionCallInputParameter[]	An array of input parameters from the flow to the action.
outputParameters	FlowActionCallOutputParameter[]	An array of output parameters from the action to the flow.

FlowActionCallInputParameter

Defines an input parameter from the flow to the action. It extends FlowBaseElement and inherits all of its fields. Available in API version 31.0 and later.

Field Name	Field Type	Description
name	string	Required. Unique name for the input parameter.
value	FlowElementReferenceOrValue	Defines the value of the input parameter.

FlowActionCallOutputParameter

Defines an output parameter from the action to the flow. It extends FlowBaseElement and inherits all of its fields. Available in API version 31.0 and later.

Field Name	Field Type	Description
assignToReference	string	Required. Specifies the variable to which you want to assign the output parameter value.
name	string	Required. Unique name for the output parameter.

FlowApexPluginCall

Defines a call to an Apex plug-in from the flow. It extends FlowNode and inherits all of its fields.

Field Name	Field Type	Description
apexClass	string	Required. The name of the Apex class.
connector	FlowConnector	Specifies which node to execute after this Apex plug-in call.
faultConnector	FlowConnector	Specifies which node to execute if the Apex plug-in call results in an error.
inputParameters	FlowApexPluginCallInputParameter[]	An array of input parameters from the flow to the Apex plug-in.
outputParameters	FlowApexPluginCallOutputParameter[]	An array of output parameters from the Apex plug-in to the flow.

FlowApexPluginCallInputParameter

Defines an input parameter from the flow to the Apex plug-in. It extends FlowBaseElement and inherits all of its fields.

Field Name	Field Type	Description
name	string	Required. Unique name for the input parameter.
value	FlowElementReferenceOrValue	Defines the value of the input parameter.

FlowApexPluginCallOutputParameter

Defines an output parameter from the Apex plug-in to the flow. It extends FlowBaseElement and inherits all of its fields.

Field Name	Field Type	Description
`assignToReference`	string	Required. Specifies the variable to which you want to assign the output parameter value.
`name`	string	Required. Unique name for the output parameter.

FlowAssignment

Defines an assignment node that can dynamically change the value of a variable in the flow. It extends FlowNode and inherits all of its fields.

Field Name	Field Type	Description
`assignmentItems`	FlowAssignmentItem[]	An array of assignment operations that is executed in the given order, starting from the index 0.
`connector`	FlowConnector	Specifies which node to execute after this assignment node.

FlowAssignmentItem

Defines an operation to apply to a variable. It extends FlowBaseElement and inherits all of its fields.

Field Name	Field Type	Description
`assignToReference`	string	Required. Reference to the variable to which you want to apply the specified operator.
`operator`	FlowAssignmentOperator (enumeration of type string)	Required. Operation to apply to the variable reference in the assignToReference field. Valid values are: • `Assign`–assigns the specified value to the variable in the assignToReference field. • `Add`–adds the specified value to the variable in the assignToReference field. • `AddItem`–adds the specified value as a new item to the variable in the assignToReference field. Supported for only variables that have a data type of Multipicklist. This operator automatically adds the semi-colon required to mark a value as a separate item. This operator is available in API version 34.0 and later. • `Subtract`–subtracts the specified value from the variable in the assignToReference field.
`value`	FlowElementReferenceOrValue	Defines the value that you want the operator to apply to the variable reference in the assignToReference field.

FlowChoice

A choice resource is a standalone choice option that you can reference or reuse throughout the flow. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`choiceText`	string	Required. Choice label to display in the screen.
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: • `Currency` • `Date` • `Number` • `String` • `Boolean`
`userInput`	FlowChoiceUserInput	Enables the choice to allow user input when the choice is selected. Not supported for choices in multi-select fields.
`value`	FlowElementReferenceOrValue	Actual value that's used during flow execution, for example, in assignments, calls to Apex plug-ins, and record elements. If null, this choice always has the value of null.

FlowChoiceUserInput

Allows the choice to include a user input field that appears when the choice is selected by the user. User input isn't supported for choices in multi-select fields. It extends FlowBaseElement and inherits all of its fields.

Field Name	Field Type	Description
`isRequired`	boolean	Indicates whether users are required to enter something into the field when they select the choice.
`promptText`	string	Text that is displayed to prompt the user for input at runtime. Supports merge fields.
`validationRule`	FlowInputValidationRule	Rule used at runtime to validate the user input.

FlowCondition

Defines a condition for a rule. It extends FlowBaseElement and inherits all of its fields.

Field Name	Field Type	Description
`leftValueReference`	string	Required. Unique name of the element that serves as the left side of the condition expression.
`operator`	FlowComparisonOperator (enumeration of type string)	Required. Valid values are: • `EqualTo`

Field Name	Field Type	Description
		• `NotEqualTo` • `GreaterThan` • `LessThan` • `GreaterThanOrEqualTo` • `LessThanOrEqualTo` • `StartsWith` • `EndsWith` • `Contains` • `IsNull` • `WasSet`—This value is available in API version 30.0 and later. • `WasSelected`—Requires a choice on the left side. • `WasVisited`—Requires a node on the left side.
`rightValue`	FlowElementReferenceOrValue	Unique name of an element or the actual value (such as text or a number) for the right side of the condition expression.

FlowConnector

Connectors determine the order in which the nodes of the flow are executed. A connector defines and links to the subsequent node. It extends FlowBaseElement and inherits all of its fields.

Field Name	Field Type	Description
`targetReference`	string	Required. Which node to execute after completing the current node.

FlowConstant

A constant resource defines a fixed value that can be used throughout your flow. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: • `Currency` • `Date` • `Number` • `String` • `Boolean`

Field Name	Field Type	Description
`value`	FlowElementReferenceOrValue	Default value of the constant. This field can't have merge fields, nor can it reference another resource besides `$GlobalConstant.EmptyString`.

FlowDecision

Decision node that evaluates a set of rules and routes the flow execution based on the first rule that evaluates to true. It extends FlowNode and inherits all of its fields.

Field Name	Field Type	Description
`defaultConnector`	FlowConnector	Specifies which node to execute if none of the rules evaluate to true.
`defaultConnectorLabel`	string	Label for the default connector.
`rules`	FlowRule[]	An array of rules for the decision. The rules are evaluated in the order they're listed, and the connector of the first true rule is used. If no rules are true, then the default connector is used. In the Cloud Flow Designer, rules are referred to as "outcomes."

FlowDynamicChoiceSet

Looks up data or metadata from an sObject and dynamically generates a set of choices at run time. It extends FlowElement and inherits all of its fields. Depending on the fields that are set, this element represents either a record choice or a picklist choice.

- A *record choice* dynamically generates choices based on records that meet specified filter criteria. If a dynamic choice doesn't have the `picklistField` and `picklistObject` parameters set, it is a record choice and can't have a data type of `Picklist` or `Multipicklist`.
- A *picklist choice* dynamically generates choices based on the available values for a picklist or multi-select picklist field. If a dynamic choice has the `picklistField` and `picklistObject` parameters set, it is a picklist choice and must have a data type of `Picklist` or `Multipicklist`.

Note: You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: • `Currency` • `Date` • `Number` • `String` • `Boolean` • `Picklist`—Picklist choices only • `Multipicklist`—Picklist choices only

Field Name	Field Type	Description
		`Picklist` and `Multipicklist` are available in API version 35.0 and later.
`displayField`	string	Required for record choices. Which field from the sObject to display to the user as choice labels. For example, for an account, use the DisplayField "Name" if you want the dynamically generated choices to be displayed as the account names from the records that are retrieved from the database. Not supported for picklist choices. Picklist choices always display the labels for the retrieved picklist values.
`filters`	FlowRecordFilter[]	An array of filters to apply to the records that are retrieved from the database. For example, you may want to filter accounts to include only those that were created in the past three months. Not supported for picklist choices.
`limit`	int	Maximum number of choices to include in the generated set of choices. Maximum and default: 200. If `sortField` and `sortOrder` are also specified, the records are sorted before the `limit` takes effect. This field is available in API version 25.0 and later.
`object`	string	Required for record choices. The sObject whose fields you want to retrieve from the database and use to generate the set of choices. For example, use "Account" to dynamically generate choices from the information in account records in the database. Not supported for picklist choices.
`outputAssignments`	FlowOutputFieldAssignment[]	An array that assigns fields from the user-selected record to variables that can be used elsewhere in the flow. For example, when the user selects an account name from the dynamically generated list of choice options, outputAssignments can assign the Id and AnnualRevenue from the user-selected account to variables that you specify. Not supported for picklist choices.
`picklistField`	string	Required for picklist choices. The field whose available values you want to retrieve from the database and use to generate the picklist choice. For example, use "Industry" to dynamically generate one choice for each available value on the Industry picklist field. Not supported for record choices.

Field Name	Field Type	Description
		This field is available in API version 35.0 and later.
`picklistObject`	string	Required for picklist choices. The sObject whose field metadata you want to retrieve from the database and use to generate the picklist choice. For example, use "Account" to dynamically generate choices from a picklist field on the Account object. Not supported for record choices. This field is available in API version 35.0 and later.
`sortField`	string	Field that is used for sorting the records that meet the filter criteria. If this field isn't specified then the returned records are not sorted. You can only sort records by fields that have the `Sort` API field property, as specified in SOAP API. Not supported for picklist choices. This field is available in API version 25.0 and later.
`sortOrder`	SortOrder (enumeration of type string)	Order in which to sort the records. If this field isn't specified, then the results are not sorted. Valid values are: • `Asc`—Ascending • `Desc`—Descending Not supported for picklist choices. This field is available in API version 25.0 and later.
`valueField`	string	Stored value for the choice, which may differ from what is displayed to the user as the choice options (`displayField`). For example, the `displayField` may be the account "Name" while the valueField is the account "Id." Not supported for picklist choices. Picklist choices always store the API value for the retrieved picklist values.

FlowElement

Base class for all flow elements. This is an abstract class. It extends FlowBaseElement and inherits all of its fields.

Field Name	Field Type	Description
`description`	string	Description of the flow element.
`name`	string	Required. Unique name of the flow element.

FlowBaseElement

Base class for all flow elements that require contextual information in metadata values. This is an abstract class. FlowBaseElement is available in API version 32.0 and later.

Field Name	Field Type	Description
`processMetadataValues`	FlowMetadataValue[]	Contextual information for the element.

FlowMetadataValue

Defines contextual information that can be passed between elements in a flow. Flow metadata values can be used in an application that produces or consumes flows. FlowMetadataValue is available in API version 32.0 and later.

Field Name	Field Type	Description
`name`	string	Required. Name for the metadata value. This name doesn't need to be unique across all elements.
`value`	FlowElementReferenceOrValue	Reference or value for the metadata value.

FlowElementReferenceOrValue

Defines a reference to an existing element or a particular value that you specify. Make sure that you specify only *one* of the fields.

Field Name	Field Type	Description
`booleanValue`	boolean	Use this field to specify a boolean value. Do not use this field if you want to specify a different data type or an element reference.
`dateTimeValue`	dateTime	Use this field to specify a dateTime value. Do not use this field if you want to specify a different data type or an element reference. This field is available in API version 30.0 and later.
`dateValue`	date	Use this field to specify a date value. Do not use this field if you want to specify a different data type or an element reference.
`elementReference`	string	Use this field to specify the name of an existing element. Do not use this field if you want to specify a value instead of an element reference.
`numberValue`	double	Use this field to specify a double value. Do not use this field if you want to specify a different data type or an element reference.
`stringValue`	string	Use this field to specify a string value. Do not use this field if you want to specify a different data type or an element reference.

FlowFormula

Calculates a value using functions and elements in the flow. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	The data type for the formula. Valid values are: • `Boolean` • `Currency` • `Date` • `DateTime` • `Number` • `String` `dataType` defaults to `Number` if it isn't defined in a formula. This field is available in API version 31.0 and later.
`expression`	string	Required. Salesforce formula expression. The return value must match the data type. See "Limitations for Flow Formulas" in the Salesforce Help. For API version 30.0 and earlier, the return value must be numeric.
`scale`	int	Scale of the return value, specifically, the number of digits to the right of the decimal point. Only supported for Currency and Number data types.

FlowInputFieldAssignment

Assigns the value for a record field based on a resource or static value. It extends FlowBaseElement and inherits all of its fields.

Note: You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
`field`	string	Required. Name of the field that is to be assigned a value while a record is being created or updated.
`value`	FlowElementReferenceOrValue	Value that is to be assigned to the field.

FlowInputValidationRule

Validation rules verify that the data entered by the user meets the specified requirements. If the validation rule evaluates to false, then the specified error message is displayed.

Field Name	Field Type	Description
`errorMessage`	string	Required. Error message to display when `formulaExpression` evaluates to false.
`formulaExpression`	string	Required. Boolean formula used to validate the user input. See "Limitations for Flow Formulas" in the Salesforce Help.

FlowLoop

A construct for iterating through a collection. It extends FlowNode and inherits all of its fields. FlowLoop is available in API version 30.0 and later.

Field Name	Field Type	Description
`nextValueConnector`	FlowConnector	Points to the element that the flow navigates to for each of the entries in the collection. This is where the flow goes for the next value in the collection.
`noMoreValuesConnector`	FlowConnector	Points to the element to navigate to when all entries in the collection have been looped through.
`collectionReference`	string	Required. The collection being looped through.
`assignNextValueToReference`	string	Required. The variable to which the current value in the collection is assigned before navigating to the target of `nextValueConnector`.
`iterationOrder`	iterationOrder (enumeration of type string)	Valid values are: • `Asc`—Iterate through the collection in the order the values are listed (first to last). • `Desc`—Iterate through the collection in the reverse order the values are listed (last to first).

FlowNode

A node is a type of element that is visible in the flow diagram. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`label`	string	Required. Name of the node. This non-unique label is different from the unique name of the node, which is inherited from FlowElement.
`locationX`	int	Required. Horizontal location of the node, in pixels from the left.
`locationY`	int	Required. Vertical location of the node, in pixels from the top.

FlowOutputFieldAssignment

Assigns an record field's value from a record to a variable that can be used elsewhere in the flow. The record may be selected by a record lookup or via a user selection for a choice. It extends FlowBaseElement and inherits all of its fields.

Note: You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
`assignToReference`	string	Required. Reference to the variable where you want to store the value of the record field.

Field Name	Field Type	Description
field	string	Required. Name of the field whose value is to be assigned after a record lookup.

FlowRecordCreate

Create a new record in the database using values from the flow. It extends FlowNode and inherits all of its properties.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
assignRecordIdToReference	string	Reference to the variable where you want to store the ID after the record is created.
connector	FlowConnector	Specifies which node to execute after creating the record.
faultConnector	FlowConnector	Specifies which node to execute if the attempt to create a record results in an error.
inputAssignments	FlowInputFieldAssignment[]	An array that assigns values to the specified fields of the record being created.
object	string	Required. sObject for the record to be created by this element
inputReference	string	Specifies the sObject variable whose field values are used to populate the new record's fields.

FlowRecordDelete

Deletes one or more records in the database. It extends FlowNode and inherits all of its fields.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
connector	FlowConnector	Specifies which node to execute after deleting the record.
faultConnector	FlowConnector	Specifies which node to execute if the attempt to delete a record results in an error.

Field Name	Field Type	Description
`filters`	FlowRecordFilter[]	An array that specifies the criteria used to select which records to delete from the database. For example, you may want to delete accounts whose last activity was older than a specified date.
`object`	string	Required. The name of the object whose records are deleted.
`inputReference`	string	Specifies the sObject variable whose record ID is used to identify which record to delete in the database.

FlowRecordFilter

Sets the criteria for searching records in the database. It extends FlowBaseElement and inherits all of its fields.

Note: You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
`field`	string	Required. The field to be used for filtering records.
`operator`	FlowRecordFilterOperator (enumeration of type string)	Required. Valid values are: • `EqualTo` • `NotEqualTo` • `GreaterThan` • `LessThan` • `GreaterThanOrEqualTo` • `LessThanOrEqualTo` • `StartsWith` • `EndsWith` • `Contains` • `IsNull`
`value`	FlowElementReferenceOrValue	Reference or value used with the field and operator to filter records.

FlowRecordLookup

Finds a record in the database and uses or stores the values from its fields in the flow. It extends FlowNode and inherits all of its fields.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
assignNullValuesIfNoRecordFound	boolean	Specifies that all values are set to `null` if the record is not found. This field is available in API version 30.0 and later.
connector	FlowConnector	Specifies which node to execute after completing the record lookup.
faultConnector	FlowConnector	Specifies which node to execute if the attempt to look up a record results in an error.
filters	FlowRecordFilter[]	An array that specifies the criteria used to select the record from the database. If the filters return more than one record, they are sorted according to the specified `sortField` and `sortOrder`. The first record in the sorted list is then selected. If either the `sortField` or `sortOrder` is not specified, then the first record returned is selected. Note, however, that records are not returned in any particular order.
limit	int	The maximum number of records to return, to limit the amount of data received. This field is available in API version 30.0 and later.
object	string	Required. Name of the sObject from which to select the record.
outputAssignments	FlowOutputFieldAssignment[]	An array that assigns fields from the selected record to variables that can be used elsewhere in the flow.
outputReference	string	Specifies the sObject variable that stores the queried fields' values.
queriedFields	string[]	An array that specifies which fields from the selected record is saved to the specified sObject variable.
sortField	string	Field that is used for sorting the records that meet the filter criteria. If this field isn't specified then the returned records are not sorted. You can only sort records by fields that have the `Sort` API field property, as specified in SOAP API. This field is available in API version 25.0 and later.
sortOrder	SortOrder (enumeration of type string)	Order in which to sort the records. If this field isn't specified, then the results are not sorted. Valid values are: • `Asc`—Ascending • `Desc`—Descending This field is available in API version 25.0 and later.

FlowRecordUpdate

Finds records in the database and updates them with values from the flow. It extends FlowNode and inherits all of its fields.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

You can't reference sObject custom fields of type Geolocation anywhere in a flow. For example, Geolocation fields can't be used in record filters, in input or output field assignments, or as display, value, or sort fields.

Field Name	Field Type	Description
`connector`	FlowConnector	Specifies which node to execute after completing the record update.
`faultConnector`	FlowConnector	Specifies which node to execute if the attempt to update a record results in an error.
`filters`	FlowRecordFilter[]	An array that specifies the criteria used to select the records to update in the database.
`inputAssignments`	FlowInputFieldAssignment[]	An array that assigns values to the specified fields of the record being updated.
`object`	string	Required. Name of the sObject whose records are updated.
`inputReference`	string	Specifies the sObject variable whose field values is used to update the record's fields.

FlowRule

Defines the conditions and logic that would enable a rule to evaluate to true. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`conditionLogic`	string	Specifies logic for the conditions. Value can be: • `and`—evaluates to true only if all of its conditions evaluate to true • `or`—evaluates to true if any of its conditions evaluate to true • Advanced logic like `1 AND (2 OR 3)`—evaluates to true if the first condition is true and either the second or third condition is true When you use advanced logic, the string must consist of 1,000 or fewer characters. This field is available in API version 33.0 and later.
`conditions`	FlowCondition[]	An array of conditions for the rule.
`connector`	FlowConnector	Specifies which node to execute if this is the first rule that evaluates to true in a decision.

Field Name	Field Type	Description
`label`	string	Required. Label for the connector.

FlowScreen

Screens provide the ability to capture information from users and display information to users. It extends FlowNode and inherits all of its fields.

Field Name	Field Type	Description
`allowBack`	boolean	Indicates whether to show (true) or hide (false) the **Previous** button on the screen at runtime. When true, the **Previous** button appears only if the user visited a previous screen in the flow path. Set this to false when revisiting the previous screen would trigger an action that should not be repeated, such as a credit card transaction. This field is available in API version 26.0 and later. Default: true You can set either `allowBack` or `allowFinish` to false, but not both.
`allowFinish`	boolean	Indicates whether to show (true) or hide (false) the **Finish** button on the screen at runtime. When true, the **Finish** button appears only if the screen element is the end of a flow path. Set this to false if you need the user to go back to a previous screen to continue or complete the flow. For example, you wouldn't want to offer a **Finish** button on a screen that tells the user to go back and make corrections on a previous screen. This field is available in API version 26.0 and later. Default: true You can set either `allowBack` or `allowFinish` to false, but not both.
`allowPause`	boolean	Indicates whether to show (true) or hide (false) the **Pause** button on the screen at runtime. A flow screen displays the **Pause** button if all of the following conditions are true. • In the organization's workflow and approvals settings, "Enable Users to Pause Flows" is enabled. • `allowPause` for the screen is set to true. • If the flow is embedded in a Visualforce page, the <flow:interview> component has its `showAllowPause` attribute set to true. This field is available in API version 33.0 and later. Default: true

Field Name	Field Type	Description
`connector`	FlowConnector	Specifies which node to execute after the screen node.
`fields`	FlowScreenField[]	An array of fields to display on the screen.
`helpText`	string	Text that appears if the end user clicks the "Help for this form" link. Supports merge fields in API version 26.0 and later.
`pausedText`	string	Confirmation message that appears when an end-user clicks **Pause**. This field is available in API version 33.0 and later.

FlowScreenField

Configurable field on a screen. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`choiceReferences`	string[]	An array of references to FlowChoices or FlowDynamicChoiceSets. The resulting choice options appear in the order specified in this array, where the element at index 0 provides the top-most choice option. Supported for the following screen field types: • RadioButtons • DropdownBox • MultiSelectCheckboxes • MultiSelectPicklist Multi-select checkboxes and multi-select picklist fields are available in API version 26.0 and later.
`dataType`	FlowDataType (enumeration of type string)	Required. Data type of this screen field. Only supported for the InputField, RadioButtons, and DropdownBox screen field types. Valid types are: • `Currency` • `Date` • `Number` • `String` • `Boolean` A boolean InputField appears as a checkbox field at runtime. Checkbox input fields are available in API version 26.0 and later.

Field Name	Field Type	Description
		Only the string data type is supported for multi-select checkboxes and multi-select picklist fields. Multi-select fields are available in API version 26.0 and later.
`defaultSelectedChoiceReference`	string	The name of the FlowChoice element to use as the default value for the screen field. Supported for the following screen field types: • RadioButtons • DropdownBox • MultiSelectCheckboxes • MultiSelectPicklist For DropdownBox field types only, if the defaultSelectedChoiceReference is empty or null, the reference at index 0 of choiceReferences are used as the default value. You can specify only one FlowChoice element as the default value for multi-select checkboxes and multi-select picklist fields. Multi-select fields are available in API version 26.0 and later.
`defaultValue`	FlowElementReferenceOrValue	The value that is used by default when this screen field requires users to provide input. Only supported for InputField, LargeTextArea, and PasswordField.
`fieldText`	string	Field label that is displayed on the screen. Supports merge fields.
`fieldType`	FlowScreenFieldType (enumeration of type string)	Required. Valid values are: • `DisplayText` • `InputField` • `LargeTextArea` • `PasswordField` • `RadioButtons` • `DropdownBox` • `MultiSelectCheckboxes` • `MultiSelectPicklist` At runtime, each multi-select field stores its field value as a concatenation of the user-selected choice values, separated by semicolons. Any semicolons in the selected choice values are removed when added to the multi-select field value.

Field Name	Field Type	Description
		Multi-select checkboxes and multi-select picklist fields are available in API version 26.0 and later.
`helpText`	string	Required. Text that appears if the end user clicks the help icon ( ) for the screen field. Supports merge fields in API version 26.0 and later.
`isRequired`	boolean	Indicates whether the user must select a choice or provide input. Not supported for DisplayText or boolean inputField.
`scale`	int	The scale of this screen field if its data type is number or currency. The scale sets the number of digits to the right of the decimal point.
`validationRule`	FlowInputValidationRule	Rule used to validate the user input when this screen field is of type InputField, LargeTextArea, or PasswordField.

FlowStep

Steps function as placeholders when you're building a flow. It extends FlowNode and inherits all of its fields.

Field Name	Field Type	Description
`connectors`	FlowConnector[]	Specifies which node to execute after the step node.

FlowSubflow

A subflow element references another flow, which it calls at run time. The flow that contains the subflow element is referred to as the master flow. FlowSubflow extends FlowNode and inherits all of its fields. It is available in API version 25.0 and later.

Field Name	Field Type	Description
`connector`	FlowConnector	Specifies which node to execute after the subflow.
`flowName`	string	References the flow to call at runtime. The value must be a unique name of a flow and can't contain an appended hyphen and version number. The referenced flow must have been created in the Cloud Flow Designer.
`inputAssignments`	FlowSubflowInputAssignment[]	An array of input variable assignments that are set at the start of the referenced flow.
`outputAssignments`	FlowSubflowOutputAssignment[]	An array of output variable assignments that are set at the end of the referenced flow.

FlowSubflowInputAssignment

Assigns an element or value from the master flow to a variable in the referenced flow. Input assignments occur when the subflow calls the referenced flow. It extends FlowBaseElement and inherits all of its fields. It is available in API version 25.0 and later.

Field Name	Field Type	Description
`name`	string	Required. Unique name for the variable in the referenced flow.
`value`	FlowElementReferenceOrValue	Defines the value to assign to the variable.

FlowSubflowOutputAssignment

Assigns the value of a variable from the referenced flow to a variable in the master flow. Output assignments occur when the referenced flow is finished running. It extends FlowBaseElement and inherits all of its fields. It is available in API version 25.0 and later.

Field Name	Field Type	Description
`assignToReference`	string	Required. Unique name for the variable in the master flow.
`name`	string	Required. Unique name for the variable in the referenced flow.

FlowTextTemplate

Defines a text template that can be used throughout the flow. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`text`	string	Actual text of the template. Supports merge fields.

FlowVariable

Variables allow you to create updatable values to use in the flow. FlowVariable extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: • `Boolean` • `Currency` • `Date` • `DateTime`—This value is available in API version 30.0 and later. • `Number` • `Multipicklist`—This value is available in API version 34.0 and later.

Field Name	Field Type	Description
		• `Picklist`—This value is available in API version 34.0 and later. • `String` • `sObject`
`isCollection`	boolean	Indicates whether the variable is a collection of values. This field is available in API version 30.0 and later. In API version 32.0 and later, a collection variable can be of any data type. Default value is `False`.
`isInput`	boolean	Indicates whether the variable can be set at the start of the flow using URL parameters, Visualforce controllers, or subflow inputs. This field is available in API version 25.0 and later. Default value: • `False` for a variable created in API version 25.0 and later or in the Cloud Flow Designer in Summer '12 and later. • `True` for a variable created in API version 24.0 or in the Cloud Flow Designer in Spring '12 and earlier. Warning: Disabling input or output access for an existing variable can break the functionality of applications and pages that call the flow and access the variable. For example, you can access variables from URL parameters, Visualforce controllers, subflows, and processes.
`isOutput`	boolean	Indicates whether the variable's value can be accessed from Visualforce controllers and other flows. This field is available in API version 25.0 and later. Default value: • `False` for a variable created in API version 25.0 and later or in the Cloud Flow Designer in Summer '12 and later. • `True` for a variable created in API version 24.0 or in the Cloud Flow Designer in Spring '12 and earlier.

Field Name	Field Type	Description
		Warning: Disabling input or output access for an existing variable can break the functionality of applications and pages that call the flow and access the variable. For example, you can access variables from URL parameters, Visualforce controllers, subflows, and processes.
`objectType`	string	Object type of this variable if its data type is `sObject`.
`scale`	int	Scale of this variable if its data type is Number or Currency.
`value`	FlowElementReferenceOrValue	Default value of this variable. Default values aren't supported if the variable's data type is `Picklist` or `Multipicklist`.

FlowWait

Waits for one or more defined events to occur. FlowWait extends FlowNode and inherits all of its fields. FlowWait is available in API version 32.0 and later.

Field Name	Field Type	Description
`defaultConnector`	FlowConnector	Specifies which node to execute if the conditions are false for every event in the Wait element.
`defaultConnectorLabel`	string	Label for the default connector.
`faultConnector`	FlowConnector	Specifies which node to execute if the attempt to wait results in an error. If any of the wait events fail, the flow takes the fault connector.
`waitEvents`	FlowWaitEvent[]	An array of events that the Wait element is waiting for. If the conditions for every event evaluate to `false`, the `defaultConnector` is used.

FlowWaitEvent

An event that a FlowWait element is waiting for. FlowWaitEvent extends FlowElement and inherits all of its fields. FlowWaitEvent is available in API version 32.0 and later.

Field Name	Field Type	Description
`conditionLogic`	string	Specifies logic for the conditions. Value can be: • `and`—evaluates to true only if all of its conditions evaluate to true • `or`—evaluates to true if any of its conditions evaluate to true • Advanced logic like `1 AND (2 OR 3)` —evaluates to true if the first condition is true and either the second or third condition is true When you use advanced logic, the string must consist of 1,000 or fewer characters. This field is available in API version 33.0 and later.
`conditions`	FlowCondition	An array of conditions that must be true for the flow to wait for this event.
`connector`	FlowConnector	Specifies which node to execute if this is the first event that occurs.
`eventType`	string	Required. The event's type. The type determines which input parameters are available to define this event. Valid values are: • `AlarmEvent`—This event is an alarm based off of an absolute date/time value. • `DateRefAlarmEvent`—This event is an alarm based off of a date/time field on a record.
`inputParameters`	FlowWaitEventInputParameter[]	An array of the event's input parameters. The parameter values are set by using values from the flow.
`label`	string	Required. Label for the wait event.
`outputParameters`	FlowWaitEventOutputParameter[]	An array of the event's output parameters. The parameter values are assigned from the event to variables in the flow.

FlowWaitEventInputParameter

An input parameter for FlowWaitEvent. The parameter's value is set by using values from the flow. It extends FlowBaseElement and inherits all of its fields. FlowWaitEventInputParameter is available in API version 32.0 and later.

Field Name	Field Type	Description
name	string	Required. Unique name for the input parameter.
value	FlowElementReferenceOrValue	Defines the value of the input parameter.

FlowWaitEventOutputParameter

An output parameter for FlowWaitEvent. The parameter's value is assigned to a variable in the flow so that it can be referenced in another part of the flow. It extends FlowBaseElement and inherits all of its fields. FlowWaitEventOutputParameter is available in API version 32.0 and later.

Field Name	Field Type	Description
assignToReference	string	Required. Specifies the variable to which you want to assign the output parameter value.
name	string	Required. Unique name for the output parameter.

Declarative Metadata Sample Definition

A sample XML definition of a flow is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Flow xmlns="http://soap.sforce.com/2006/04/metadata">
    <choices>
        <name>Bad</name>
        <choiceText>Bad</choiceText>
        <dataType>Number</dataType>
        <value>
            <numberValue>0.0</numberValue>
        </value>
    </choices>
    <choices>
        <name>Excellent</name>
        <choiceText>Excellent</choiceText>
        <dataType>Number</dataType>
        <value>
            <numberValue>20.0</numberValue>
        </value>
    </choices>
    <choices>
        <name>Fair</name>
```

```
        <choiceText>Fair</choiceText>
        <dataType>Number</dataType>
        <value>
            <numberValue>10.0</numberValue>
        </value>
    </choices>
    <choices>
        <name>Good</name>
        <choiceText>Good</choiceText>
        <dataType>Number</dataType>
        <value>
            <numberValue>15.0</numberValue>
        </value>
    </choices>
    <description>Simple Flow app to calculate a Tip according to corporate
        policies</description>
    <formulas>
        <name>fTipAmount</name>
        <dataType>Number</dataType>
        <expression>{!Bill_Amount} * {!Service_Quality} / 100</expression>
        <scale>0</scale>
    </formulas>
    <formulas>
        <name>fTotalAmount</name>
        <dataType>Number</dataType>
        <expression>{!fTipAmount} + {!Bill_Amount}</expression>
        <scale>0</scale>
    </formulas>
    <label>Tip Calculator App</label>
    <screens>
        <name>Simple_Tip_Calculator</name>
        <label>Simple Tip Calculator</label>
        <locationX>513</locationX>
        <locationY>112</locationY>
        <connector>
            <targetReference>TipAmount</targetReference>
        </connector>
        <fields>
            <name>Bill_Amount</name>
            <dataType>Currency</dataType>
            <fieldText>Bill Amount</fieldText>
            <fieldType>InputField</fieldType>
            <isRequired>false</isRequired>
            <scale>2</scale>
        </fields>
        <fields>
            <name>Service_Quality</name>
            <choiceReferences>Excellent</choiceReferences>
            <choiceReferences>Good</choiceReferences>
            <choiceReferences>Fair</choiceReferences>
            <choiceReferences>Bad</choiceReferences>
            <dataType>Number</dataType>
            <fieldText>Service Quality</fieldText>
            <fieldType>RadioButtons</fieldType>
```

```
            <isRequired>false</isRequired>
            <scale>2</scale>
        </fields>
    </screens>
    <screens>
        <name>TipAmount</name>
        <label>Tip Amount</label>
        <locationX>518</locationX>
        <locationY>266</locationY>
        <fields>
            <name>TipSUmmary</name>
            <fieldText>&lt;TEXTFORMAT
                LEADING=&quot;2&quot;&gt;&lt;P
                ALIGN=&quot;LEFT&quot;&gt;&lt;FONT
                FACE=&quot;Arial&quot;
                STYLE=&quot;font-size:12px&quot;
                COLOR=&quot;#000000&quot;
                LETTERSPACING=&quot;0&quot;
                KERNING=&quot;0&quot;&gt;If you think the quality of
                service is &lt;FONT
                KERNING=&quot;1&quot;&gt;{!Service_Quality},
                &lt;/FONT&gt;for a meal of {!Bill_Amount} you should tip
                {!fTipAmount}, so the total recommended amount should be
                {!fTotalAmount}&lt;/FONT&gt;&lt;/P&gt;&lt;/TEXTFORMAT&gt;</fieldText>
            <fieldType>DisplayText</fieldType>
        </fields>
    </screens>
    <startElementReference>Simple_Tip_Calculator</startElementReference>
</Flow>
```

FlowDefinition

Represents the flow definition's description and active flow version number.

To activate a flow, modify the metadata object and set the `activeVersionNumber` to the version number to activate. To deactivate an active flow version, set the `activeVersionNumber` to 0 (zero) or omit the value.

Declarative Metadata File Suffix and Directory Location

FlowDefinitions are stored in the `flowDefinitions` directory of the corresponding package directory. The file name matches the flow definition's unique full name, and the extension is `.flowDefinition`.

Version

FlowDefinition is available in API version 34.0 and later.

Field Name	Field Type	Description
`activeVersionNumber`	int	The version number of the active flow.
`description`	string	Description of the flow definition.

Field Name	Field Type	Description
masterLabel	string	Label for the flow definition.

Folder

Represents a folder. This type extends the Metadata metadata type and inherits its fullName field. Four folder types currently exist in Salesforce:

- Document folder
- Email folder
- Report folder
- Dashboard folder

Folder type names end with the "Folder" suffix. For example, the type name of an email folder is "EmailFolder".

File Suffix and Directory Location

Folders are stored in the corresponding component directory of the package. These directories are named documents, email, reports, and dashboards. Folders do not have a text file representation—they are containers for files. For each folder, an accompanying metadata file named *FolderName*-meta.xml is created at the same directory level. The *FolderName*-meta.xml metadata file contains the metadata information for that folder, such as the accessType. For example, for a documents folder named sampleFolder, there's a *sampleFolder-meta.xml* within the documents folder of the package.

Version

Folders are available in API version 11.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
accessType	FolderAccessTypes (enumeration of type string)	Required. The type of access for this folder. Valid values are: • Shared. This folder is accessible only by the specified set of users. • Public. This folder is accessible by all users, including portal users. • PublicInternal. This folder is accessible by all users, excluding portal users. This setting is available for report and dashboard folders in organizations with a partner portal or Customer Portal enabled. • Hidden. This folder is hidden from all users.
fullName	string	The name used as a 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 end with an

Field Name	Field Type	Description
		underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`name`	string	Required. The name of the document folder.
`publicFolderAccess`	PublicFolderAccess (enumeration of type string)	If `Public` is the value for `accessType`, this field indicates the type of access all users will have to the contents of the folder. Valid values include: • `ReadOnly`. All users can read the contents of the folder, but no user can change the contents. • `ReadWrite`. All users can read or change the contents of the folder.
`sharedTo`	SharedTo	Sharing access for the folder. See "Sharing Considerations" in the Salesforce online help.

Declarative Metadata Sample Definition

The following is the package manifest definition of a document folder that contains a document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>basic</fullName>
    <types>
        <members>sampleFolder</members>
        <members>sampleFolder/TestDocument.txt</members>
        <name>Document</name>
    </types>
    <version>36.0</version>
</Package>
```

The following is an example of the `sampleFolder-meta.xml` metadata file for the sampleFolder document folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentFolder xmlns="http://soap.sforce.com/2006/04/metadata">
    <accessType>Public</accessType>
    <name>sampleFolder</name>
    <publicFolderAccess>ReadWrite</publicFolderAccess>
</DocumentFolder>
```

SEE ALSO:

- Dashboard
- Document
- EmailTemplate
- Report

FolderShare

Represents the settings for enhanced analytics folder sharing. Users can control access to reports or dashboards by giving others Viewer, Editor or Manager access to the folder that contains the report or dashboard.

File Suffix and Directory Location

FolderShare objects are stored in the `reports` and `dashboards` directories. For each report or dashboard folder it contains, there is a metadata file named *`FolderName`*`-meta.xml`. The *`FolderName`*`-meta.xml` metadata file contains the metadata information for that folder, such as the `accessLevel`. For example, if the `reports` directory contains a reports folder named `myReportsFolder`, it also has a *`myReportsFolder-meta.xml`* file at the same level as `myReportsFolder`.

Version

FolderShare components are available in API version 28 and later.

Fields

Field Name	Field Type	Description
`accessLevel`	FolderShareAccessLevel (enumeration of type string)	Required. Specifies the combination of actions that can be taken on the folder. Valid values are: • `View`. User can run a report or refresh a dashboard, but can't edit them. All users have at least Viewer access to report and dashboard folders that have been shared with them. (Some users may have administrative permissions that give them greater access.) • `EditAllContents`. Users can view and modify the reports or dashboards in the folder, and move them to and from any other folders that they have equivalent access to. • `Manage`. Users can do everything Viewers and Editors can do, plus control other users' access to a folder.
`sharedTo`	string	Required. Specifies the user, group, or role that has the specified access level to the folder.
`sharedToType`	FolderSharedToType(enumeration of type string)	Required. Specifies the type of entity that the folder is shared with. Valid values are: • `Group`. Users in a specified public group have the specified access level to the folder. • `Manager`. Available in API version 29.0 and later. • `ManagerAndSubordinatesInternal`. Available in API version 29.0 and later. • `Role`. Users with a specified role have the specified access level to the folder. • `RoleAndSubordinates`. Users with a specified role, and users with a role subordinate to that role, have the specified access level to the folder.

Field Name	Field Type	Description
		• `RoleAndSubordinatesInternal`. Users with a specified role and users with a role subordinate to that role, except public portal users, have the specified access level to the folder. • `Organization`. All internal users have the specified access level to the folder. • `Territory`. Users in a specified territory have the specified access level to the folder. • `TerritoryAndSubordinates`. Users in a specified territory, and users in territories subordinate to that, have the specified access level to the folder. • `AllPrmUsers`. All PRM Portal users have the specified level of access to the folder. • `User`. The specified individual user has the specified level of access to the folder. • `PartnerUser`. The specified individual user of a partner portal has the specified level of access to the folder. • `AllCspUsers`. All Customer Success Portal users have the specified level of access to the folder. • `CustomerPortalUser`. The specified individual user of a customer portal has the specified level of access to the folder. • `PortalRole`. Users with a specified role in a portal have the specified access level to the folder. • `PortalRoleAndSubordinates`. Portal users with a specified role, and portal users with a role subordinate to that role, have the specified access level to the folder.

Declarative Metadata Sample Definition

The following is an example of a FolderShare component for a dashboard folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<DashboardFolder xmlns="http://soap.sforce.com/2006/04/metadata">
  <folderShares>
        <accessLevel>View</accessLevel>
        <sharedTo>R1</sharedTo>
        <sharedToType>Role</sharedToType>
    </folderShares>
</DashboardFolder>
```

This is an example of a FolderShare component for a report folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<ReportFolder xmlns="http://soap.sforce.com/2006/04/metadata">
  <folderShares>
       <accessLevel>View</accessLevel>
        <sharedTo>R1</sharedTo>
```

```
        <sharedToType>Role</sharedToType>
    </folderShares>
</ReportFolder>
```

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 (`true`) or do not have access (`false`) 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 `fullName` 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>
```

HomePageComponent

Represents the metadata associated with a home page component. You can customize the Home tab to include components such as sidebar links, a company logo, a dashboard snapshot, or custom components that you create. For more information, see "Customizing Home Tab Page Layouts" in the Salesforce online help. This type extends the Metadata metadata type and inherits its `fullName` field. Use to create, update, or delete home page component definitions.

Declarative Metadata File Suffix and Directory Location

The file suffix for home page components is `.homePageComponent` and components are stored in the `homepagecomponents` directory of the corresponding package directory.

Version

Home page components are available in API version 12.0 and later.

HomePageComponent

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

Field Name	Field Type	Description
`body`	string	If this is an HTML page component, this is the body of the HTML.
`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.
`height`	int	Required for Visualforce Area components. Indicates the height (in pixels) of the component. This field is available in API version 31.0 and later.
`links`	string[]	If the `pageComponentType` is `links`, then zero or more names of custom page links can be specified. • `ObjectWebLink` • `CustomPageWebLink`
`page`	string	This field is only available for Visualforce Area components and indicates the API name of the Visualforce page that is associated with the component. This field is available in API version 31.0 and later.

Field Name	Field Type	Description
`pageComponentType`	PageComponentType (enumeration of type string)	Required. Valid values are: • `links` • `htmlArea` • `imageOrNote` • `visualforcePage` (This value is available in API version 31.0 and later.)
`showLabel`	boolean	This field is only available for Visualforce Area components and specifies whether the component displays with a label (`true`) or not (`false`). This field is available in API version 31.0 and later.
`showScrollbars`	boolean	This field is only available for Visualforce Area components and specifies whether the component displays with scrollbars (`true`) or not (`false`). This field is available in API version 31.0 and later.
`width`	PageComponentWidth (enumeration of type string)	This field is only available for HTML and Visualforce Area components, and indicates whether this is a narrow or wide home page component. Valid values are: • `narrowComponents` • `wideComponents`

Declarative Metadata Sample Definition

The following is the definition of a home page component. See HomePageLayout and WebLink for related samples.

```
<?xml version="1.0" encoding="UTF-8"?>
<HomePageComponent xmlns="http://soap.sforce.com/2006/04/metadata">
  <height>200</height>
  <page>MyVisualforcePage</page>
  <pageComponentType>visualforcePage</pageComponentType>
  <showLabel>true</showLabel>
  <showScrollbars>true</showScrollbars>
  <width>wideComponents</width>
</HomePageComponent>
```

SEE ALSO:

HomePageLayout

WebLink

HomePageLayout

Represents the metadata associated with a home page layout. You can customize home page layouts and assign the layouts to users based on their user profile. For more information, see "Customizing Home Tab Page Layouts" in the Salesforce online help.

File Suffix and Directory Location

Home page layouts are stored in the `homePageLayouts` directory of the corresponding package directory. The extension is `.homePageLayout`.

Version

Home page components are available in API version 12.0 and later. This type extends the Metadata metadata type and inherits its `fullName` field.

Fields

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

Field Name	Field Type	Description
`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.
`narrowComponents`	string[]	The list of elements in the narrow column on the left side of the home page.
`wideComponents`	string[]	The list of elements in the wide column on the right side of the home page.

Declarative Metadata Sample Definition

The following is the definition of a home page layout. See HomePageComponent on page 370 and WebLink on page 261 for related samples.

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

SEE ALSO:

HomePageComponent

WebLink

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
`versionNumber`	string	The version number of the package. This has the format *`majorNumber.minorNumber.patchNumber`* (for example, 2.1.3).
`password`	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>
```

KeywordList

Represents a list of keywords used in community moderation. This keyword list is a type of moderation criteria that defines offensive language or inappropriate content that you don't want in your community. This type extends the Metadata metadata type and inherits its `fullName` field.

Keep the following things in mind when creating keyword list criteria:

- Your organization can have up to 30 keyword list criteria. This limit is per organization, not per community.
- A keyword list can have up to 2,000 keywords.
- Capitalization and trailing punctuation are ignored when matching your keywords to user-generated content. For example, if your criteria includes *`BadWord`*, it's matched when a user types *`BADWORD`* or *`badword.`*

File Suffix and Directory Location

KeywordList components have the suffix `.keywords` and are stored in the `moderation` directory of the corresponding package directory. The file name format follows *`community_name.keyword_list_developer_name`*`.keywords`.

Version

KeywordList components are available in API version 36.0 and later.

Special Access Rules

To view, create, edit, and delete a keyword list, you need the Manage Communities or Create and Set Up Communities permission.

Fields

Field Name	Field Type	Description
`Description`	string	A description of the keyword list.
`keywords`	Keyword[]	The keywords you want moderate in your community.
`masterLabel`	string	Required. Label for the keyword list.

Keyword

Keywords in the keyword list.

Field Name	Field Type	Description
`keyword`	string	Required. Keywords you want to moderate. • Keywords can be up to 100 characters and can include letters, numbers, spaces, and special characters. • Wildcard characters aren't supported.

Declarative Metadata Sample Definition

The following is an example of a KeywordList component.

```
<?xml version="1.0" encoding="UTF-8"?>
<KeywordList xmlns="http://soap.sforce.com/2006/04/metadata">
   <masterLabel>Bad Word List</masterLabel>
   <description>List of bad words updated by Joe in Nov 2015.</description>
   <keywords>
     <keyword>bad-word</keyword>
   </keywords>
   <keywords>
     <keyword>b a d w o r d</keyword>
   </keywords>
```

```
    <keywords>
      <keyword>b@dword</keyword>
    </keywords>
</KeywordList>
```

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>
      <name>KeywordList</name>
      <members>community1.badword_list</members>
    </types>
    <version>36.0</version>
</Package>
```

Layout

Represents the metadata associated with a page layout. For more information, see "Page Layouts" in the Salesforce online help. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: To edit the Ideas layout, 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'.

Field Name	Field Type	Description
customConsoleComponents	CustomConsoleComponents	Represents custom console components (Visualforce pages, lookup fields, or related lists; 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, <excludeButtons>Delete</excludeButtons> excludes the **Delete** button from this layout.
feedLayout	FeedLayout	Represents the values that define the feed view of a feed-based page layout. Feed-based layouts are available on Account, Case, Contact, Lead, Opportunity, and custom objects. They include a feed view and a detail view.
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 on Records" in the Salesforce online help. Valid string values are: • PersonalTagging—tag is private to user. • PublicTagging—tag is viewable 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 fields are otherwise similar to miniLayoutFields miniLayout.
platformActionList	PlatformActionList	The list of actions, and their order, that display in the Salesforce1 action bar for the layout. This field is available in API version 34.0 and later.
quickActionList	QuickActionList	The list of quick actions that display in the full Salesforce site for the page layout. This field is available in API version 28.0 and later.
relatedContent	RelatedContent	The Related Content section of the page layout. This field is available in API version 29.0 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 objects are foreign key fields on the object for the layout. For more information, see "Choose Related Objects for the Agent Console's Mini View" in the Salesforce 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, lookup fields, or related lists; 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, lookup field, or related lists; 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.
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, lookup field, or related lists; 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 field determines the unit of measurement, in pixels or percent.
isContainerAutoSizeEnabled	boolean	Required. If set to true, stacked console components in the sidebars autosize vertically. Set to true by default for newly created console components. Available in API version 32.0 and later.

Field Name	Field Type	Description
`region`	string	Required. The location of the components' container. Valid values include: • `right` • `left` • `top` • `bottom`
`sidebarComponents`	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: • `stack`—a content area with multiple frames. • `tabs`—a single content area with a list of multiple panels. • `accordian`—a collapsible content area.
`unit`	string	Required. The unit of measurement, in pixels or percent, for the height or width of the components' container. 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.
`width`	int	Required for components with a location of right or left. The width of the components' container. The `unit` field determines the unit of measurement, in pixels or percent.

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
`componentType`	string	Specifies the component type. Valid values are `KnowledgeOne`, `Lookup`, `Milestones`, `RelatedList`, `Topics`, `Files`, and `CaseExperts`. This field is available in API version 31.0 and later. The `Files` and `CaseExperts` values are available in API version 32.0 and later. Note: Case Experts is available through a pilot program.
`height`	int	Required for components with a location of top or bottom. The height of the component in the container. The `unit` field determines the unit of measurement, in pixels or percent.

Field Name	Field Type	Description
`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 accordion.
`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.
`relatedlists`	RelatedList[]	If the component is a related list, the name of the list. This field is available in API version 31.0 and later.
`unit`	string	The unit of measurement, in pixels or percent, for the height or width of the component in the container. 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.
`width`	int	Required for components with a location of right or left. The width of the component in the container. The `unit` field determines the unit of measurement, in pixels or percent.

RelatedList

Represents related list custom components on the sidebars of the Salesforce console. Available in API version 31.0 and later.

Field Name	Field Type	Description
`hideOnDetail`	boolean	If set to `true`, the related list is hidden from detail pages where it appears as a component to prevent duplicate information from showing.
`name`	string	The name of the component as it appears to console users.

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, lookup field, or related lists; 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.
`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.

FeedLayout

Represents the values that define the feed view of a feed-based page layout. Feed-based layouts are available on Account, Case, Contact, Lead, Opportunity, and custom objects. They include a feed view and a detail view. Available in API version 30.0 and later.

Field Name	Field Type	Description
`autocollapsePublisher`	boolean	Specifies whether the publisher is automatically collapsed when the page loads (`true`) or not (`false`).
`compactFeed`	boolean	Specifies whether the feed-based page layout uses a compact feed (`true`) or not (`false`). If set to `true`, feed items on the page are collapsed by default, and the feed view has an updated design.
`feedFilterPosition`	FeedLayoutFilterPosition (enumeration of type string)	Where the feed filters list is included in the layout. Valid values are: • `centerDropDown`—as a drop-down list in the center column. • `leftFixed`—as a fixed list in the left column. • `leftFloat`—as a floating list in the left column.
`feedFilters`	FeedLayoutFilter[]	The individual filters displayed in the feed filters list.
`fullWidthFeed`	boolean	Specifies whether the feed expands horizontally to take up all available space on the page (`true`) or not (`false`).
`hideSidebar`	boolean	Specifies whether the sidebar is hidden (`true`) or not (`false`).
`leftComponents`	FeedLayoutComponent[]	The individual components displayed in the left column of the feed view.
`rightComponents`	FeedLayoutComponent[]	The individual components displayed in the right column of the feed view.

FeedLayoutComponent

Represents a component in the feed view of a feed-based page layout. Available in API version 30.0 and later.

Field Name	Field Type	Description
`componentType`	FeedLayoutComponentType (enumeration of type string)	Required. The type of component. Valid values are: • `HelpAndToolLinks`—icons that link to the help topic for the page, the page layout, and, the printable view of the page. Available only on Case layouts. • `CustomButtons`—a custom button. • `Following`—an icon that toggles between a Follow button (if the user viewing a record doesn't yet follow it) and a Following indicator (if the user viewing a record does follow it). • `Followers`—a list of users who follow the record. • `CustomLinks`—a custom link. • `Milestones`—the milestone tracker, which lets users see the status of a milestone on a case. Available only on Case layouts.

Field Name	Field Type	Description
		• `Topics`—a list of topics related to the record. • `Visualforce`—a custom Visualforce component.
`height`	int	The height, in pixels, of the component. Doesn't apply to `standardComponents`
`page`	string	The name of a Visualforce page being used as a custom component.

FeedLayoutFilter

Represents a feed filter option in the feed view of a feed-based page layout. A filter must have only `standardFilter` or `feedItemType` set. Available in API version 30.0 and later.

Field Name	Field Type	Description
`feedFilterName`	string	The name of a CustomFeedFilter component. Names are prefixed with the name of the parent object. For example, *`Case.MyCustomFeedFilter`*.
`feedFilterType`	FeedLayoutFilterType (enumeration of type string)	The type of filter. Valid values are: • `AllUpdates`—shows all feed items on a record. • `FeedItemType`—shows feed items only for a particular type of activity on the record.
`feedItemType`	FeedItemType (enumeration of type string)	The type of feed item to display. Valid values are: • `ActivityEvent`—feed items related to activity on tasks and events associated with a case. Available only on Case layouts. • `AdvancedTextPost`–feed items related to group announcements posted on a feed. This value is available in API version 31.0 and later. • `AnnouncementPost`–Not used. • `ApprovalPost`—feed items related to approvals that are submitted on a feed. • `AttachArticleEvent`—feed items for activity related to attaching articles to cases. Available only on Case layouts. • `BasicTemplateFeedItem`—Not used. • `CallLogPost`—feed items for activity from the Log a Call action. Available only on layouts for objects that support Activities (tasks and events). • `CanvasPost`—feed items related to posts a canvas app makes on a feed. • `CaseCommentPost`—feed items for activity from the Case Note action. Available only on Case layouts.

Field Name	Field Type	Description
		• `ChangeStatusPost`—feed items for activity from the Change Status action. Available only on Case layouts. • `ChatTranscriptPost`—feed items for activity related to attaching Live Agent chat transcripts to cases. Available only on Case layouts. • `CollaborationGroupCreated`—feed items related to creating a public group. • `CollaborationGroupUnarchived`—Not used. • `ContentPost`—feed items related to attaching a file to a post. • `CreatedRecordEvent`—feed items related to creating a record from the publisher. • `DashboardComponentSnapshot`—feed items related to posting a dashboard snapshot on a feed. • `EmailMessageEvent`—feed items for activity from the Email action. Available only on Case layouts. • `FacebookPost`—Not used. • `LinkPost`—feed items related to attaching a URL to a post. • `MilestoneEvent`—feed items for changes to the milestone status on a case. Available only on Case layouts. • `PollPost`—feed items related to posting a poll on a feed. • `ProfileSkillPost`—feed items related to skills added to a user's Chatter profile. This value is available in API version 31.0 and later. • `QuestionPost`—feed items related to posting a question on a feed. This value is available in API version 31.0 and later. • `ReplyPost`—feed items for activity from the Portal action. Available only on Case layouts. • `RypplePost`—feed items related to creating a Thanks badge in Work.com. • `SocialPost`—feed items for activity on Twitter from the Social Post action. • `TextPost`—feed items for creating a text post from the publisher. • `TrackedChange`—feed items related to a change or group of changes to a tracked field. • `UserStatus`—Not used.

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 `customLabel` flag.
`layoutColumns`	LayoutColumn[]	The columns of the layout, depending on the style. 1, 2, or 3 columns, ordered left to right, are possible.
`style`	LayoutSectionStyle (enumeration of type string)	The style of the layout: • `TwoColumnsTopToBottom` - Two columns, tab goes top to bottom • `TwoColumnsLeftToRight` - Two columns, tab goes left to right • `OneColumn` - One column • `CustomLinks` - 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).

Field Name	Field Type	Description
`reserved`	string	This field is reserved for Salesforce. 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 values set: component, customLink, field, scontrol, page, analyticsCloudComponent, or reportChartComponent.

Field Name	Field Type	Description
`behavior`	UiBehavior (enumeration of type string)	Determines the field behavior. Valid string values: • `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
`canvas`	string	Reference to a canvas app. This field is available in API version 31.0 and later.
`component`	string	Reference to a component. Value must be *`sfa:socialCard`*. This field is available in API version 30.0 and later. This is only allowed inside a `RelatedContentItem`. *`sfa:socialCard`* is only supported on page layouts for contacts, accounts, and leads.
`customLink`	string	The `customLink` reference. This is only allowed inside a `CustomLink 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 `Description` or `MyField__c`.
`height`	int	For s-control and pages only, the height in pixels.
`page`	string	Reference to a Visualforce page.
`analyticsCloudComponent`	AnalyticsCloudComponentLayoutItem	Refers to a Wave Analytics dashboard that you can add to a standard or custom object page. This field is available in API version 34.0 and later.
`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.

Field Name	Field Type	Description
showLabel	boolean	For s-control and pages only, whether to show the label.
showScrollbars	boolean	For s-control and pages only, whether 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%.

AnalyticsCloudComponentLayoutItem

Represents the settings for a Wave Analytics dashboard on a standard or custom page. Available in API version 34.0 and later.

Field Name	Field Type	Description
assetType	string	Required. Specifies the type of Wave Analytics asset to add. The available asset type is dashboard.
devName	string	Required. Unique development name of the dashboard to add.
error	string	Error string; only populated if an error occurred in the underlying dashboard.
filter	string	Communicates initial dashboard filters for mapping data fields in the dashboard to the object's fields, so that the dashboard shows only the data that's relevant for the record being viewed.
height	int	Specifies the height of the dashboard, in pixels. The default is 400.
hideOnError	boolean	Controls whether users see a dashboard that has an error. When this attribute is set to true, if the dashboard has an error, the dashboard doesn't appear on the page. When set to false, the dashboard appears but doesn't show any data except the error. An error can happen when a user doesn't have access to Wave Analytics or to the dashboard. The default is true.
showTitle	boolean	If true, includes the dashboard's title above the dashboard. If false, the dashboard appears without a title. The default is true.
width	string	Specifies the width of the dashboard, 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%. The default is 100%.

ReportChartComponentLayoutItem

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

Field Name	Field Type	Description
`cacheData`	boolean	Indicates whether to use cached data when displaying the chart. When the attribute is set to `true`, data is cached for 24 hours. If the attribute is set to `false`, the report is run every time the page is refreshed. This field is available in API version 29.0 and later.
`contextFilterableField`	string	Unique development name of the field by which a report chart is filtered to return data relevant to the page. If set, the ID field for the parent object of the page or report type is the chart data filter. The parent object for the report type and the page must match for a chart to return relevant data.
`error`	string	Error string; only populated if an error occurred in the underlying report. This field is available in API version 31.0 and later.
`hideOnError`	boolean	Controls whether users see a chart that has an error. When there's an error and this attribute is not set, the chart doesn't show any data except the error. An error can happen for many reasons, such as when a user doesn't have access to fields used by the chart or a chart has been removed from the report. Set the attribute to `true` to hide the chart from a page on error. This field is available in API version 29.0 and later.
`includeContext`	boolean	If `true`, 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 `true`, 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: • `SMALL` • `MEDIUM` • `LARGE`

PlatformActionList

PlatformActionList represents the list of actions, and their order, that display in the Salesforce1 action bar for the layout. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionListContext`	PlatformActionListContext (enumeration of type string)	Required. The context of the action list. Valid values are: • `BannerPhoto` • `Chatter` • `Dockable` • `FeedElement`

Field Name	Field Type	Description
		• `FlexiPage` • `Global` • `ListView` • `ListViewDefinition` • `ListViewRecord` • `Lookup` • `MruList` • `MruRow` • `ObjectHomeChart` • `Photo` • `Record` • `RecordEdit` • `RelatedList` • `RelatedListRecord`
`platformActionListItems`	PlatformActionListItem[]	The actions in the PlatformActionList.
`relatedSourceEntity`	string	When the `ActionListContext` is RelatedList or RelatedListRecord, this field represents the API name of the related list to which the action belongs.

PlatformActionListItem

PlatformActionListItem represents an action in the PlatformActionList. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionName`	string	The API name for the action in the list.
`actionType`	PlatformActionType (enumeration of type string)	The type of action. Valid values are: • `ActionLink`—An indicator on a feed element that targets an API, a web page, or a file, represented by a button in the Salesforce Chatter feed UI. • `CustomButton`—When clicked, opens a URL or a Visualforce page in a window or executes JavaScript. • `InvocableAction` • `ProductivityAction`—Productivity actions are predefined by Salesforce and are attached to a limited set of objects. You can't edit or delete productivity actions. • `QuickAction`—A global or object-specific action. • `StandardButton`—A predefined Salesforce button such as New, Edit, and Delete.

Field Name	Field Type	Description
`sortOrder`	int	The placement of the action in the list.
`subtype`	string	The subtype of the action. For quick actions, the subtype is `QuickActionType`. For custom buttons, the subtype is `WebLinkTypeEnum`. For action links, subtypes are `Api`, `ApiAsync`, `Download`, and `Ui`. Standard buttons and productivity actions have no subtype.

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.

RelatedContent

RelatedContent represents the Mobile Cards section of the page layout. Available in API version 29.0 and later.

Field Name	Field Type	Description
`relatedContentItems`	RelatedContentItem[]	A list of layout items in the Mobile Cards section of the page layout.

RelatedContentItem

RelatedContentItem represents an individual item in the RelatedContentItem list. Available in API version 29.0 and later.

Field Name	Field Type	Description
`layoutItem`	LayoutItem	An individual LayoutItem in the Mobile Cards section.

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 "Define Custom Buttons and Links" in the Salesforce 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 `Fax`, `Mobile`, and `Home Phone` fields are retrieved as Phone2, Phone3, and Phone4, respectively.
`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 `sortField` is set, the `sortOrder` field determines the sort order. • `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 pane, between 1 and 4 (inclusive).
`sizeY`	int	Required. Number of rows in each column, 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: • 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 `customLink` 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 `sizeX`.
`posY`	int	Required. The item's row position in the highlights panel grid. Must be within the range of `sizeY`.
`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>
        <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>
            <layoutItems>
                <height>200</height>
                <page>mcanvasPage</page>
                <showLabel>true</showLabel>
                <showScrollbars>false</showScrollbars>
```

```
            <width>100%</width>
        </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>
 <quickActionList>
    <quickActionListItems>
        <quickActionName>FeedItem.TextPost</quickActionName>
    </quickActionListItems>
    <quickActionListItems>
        <quickActionName>FeedItem.ContentPost</quickActionName>
    </quickActionListItems>
    <quickActionListItems>
        <quickActionName>FeedItem.LinkPost</quickActionName>
    </quickActionListItems>
    <quickActionListItems>
        <quickActionName>FeedItem.PollPost</quickActionName>
    </quickActionListItems>
</quickActionList>
<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>
```

Letterhead

Represents formatting options for the letterhead in an email template. Letterheads define the look and feel of your HTML email templates. Your HTML email templates can inherit the logo, color, and text settings from a letterhead. For more information, see "Create a Letterhead" in the Salesforce online help. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

The file suffix for letterheads is `.letter` and components are stored in the `letterhead` directory of the corresponding package directory.

Version

Letterheads are available in API version 12.0 and later.

Fields

With the exception of logo, and horizontal and vertical alignment, all of these fields are required.

Field Name	Field Type	Description
`available`	boolean	Required. Indicates whether this letterhead can be used (`true`) or not (`false`), for example, in an email template.
`backgroundColor`	string	Required. The background color, in hexadecimal, for example `#FF6600`.
`bodyColor`	string	Required. The body color in hexadecimal.
`bottomLine`	LetterheadLine (enumeration of type string)	Required. The style for the bottom line. Valid style values include: • `color`. The color of the line in hexadecimal, as a string value. • `height`. The height of the line, as an int value.
`description`	string	Text description of how this letterhead differs from other letterheads.
`fullName`	string	The internal name of the letterhead, based on the `name`, but with white spaces and special characters escaped out for validity.
`footer`	LetterheadHeaderFooter	Required. The style for the footer.
`header`	LetterheadHeaderFooter	Required. The style for the header.
`middleLine`	LetterheadLine	Required. The style for the middle border line in your letterhead. Valid style values include: • `color`. The color of the line in hexadecimal, as a string value. • `height`. The height of the line, as an int value.
`name`	string	Required. The name of the letterhead.
`topLine`	LetterheadLine	Required. The style for the top horizontal line below the header. Valid style values include: • `color`. The color of the line in hexadecimal, as a string value. • `height`. The height of the line, as an int value.

LetterheadHeaderFooter

LetterheadHeaderFooter represents the properties of a header or footer.

Field	Field Type	Description
backgroundColor	string	Required. The background color of the header or footer in hexadecimal format.
height	DashboardComponent[]	Required. The height of the header or footer.
horizontalAlignment	LetterheadHorizontalAlignment (enumeration of type string)	The horizontal alignment of the header or footer. Valid values are: • None • Left • Center • Right
logo	string	The logo which is a reference to a document, for example MyFolder/MyDocument.gif.
verticalAlignment	LetterheadVerticalAlignment (enumeration of type string)	The vertical alignment of the header or footer. Valid values are: • None • Top • Middle • Bottom

LetterheadLine

LetterheadLine represents the properties of a line.

Field	Field Type	Description
color	string	Required. The color of the line in hexadecimal format.
height	int	Required. The height of the line.

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<Letterhead xmlns="http://soap.sforce.com/2006/04/metadata">
    <available>true</available>
    <backgroundColor>#CCCCCC</backgroundColor>
    <bodyColor>#33FF33</bodyColor>
    <bottomLine>
        <color>#3333FF</color>
        <height>5</height>
    </bottomLine>
    <description>INITIAL</description>
    <footer>
        <backgroundColor>#FFFFFF</backgroundColor>
        <height>100</height>
```

```
        <horizontalAlignment>Left</horizontalAlignment>
        <verticalAlignment>Top</verticalAlignment>
    </footer>
    <header>
        <backgroundColor>#FFFFFF</backgroundColor>
        <height>100</height>
        <horizontalAlignment>Left</horizontalAlignment>
        <verticalAlignment>Top</verticalAlignment>
    </header>
    <middleLine>
        <color>#AAAAFF</color>
        <height>5</height>
    </middleLine>
    <name>SimpleLetterheadLabel</name>
    <topLine>
        <color>#FF99FF</color>
        <height>5</height>
    </topLine>
</Letterhead>
```

LiveChatAgentConfig

Represents the configuration of an organization's Live Agent deployment, such as how many chats can be assigned to an agent and whether or not chat sounds are enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveChatAgentConfig configurations are referenced in the `<developer_name>.liveChatAgentConfig` file in the `liveChatAgentConfigs` directory.

Version

LiveChatAgentConfig is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`assignments`	AgentConfigAssignments	Specifies how agent configurations are assigned to Live Agent users. Agent configurations can be assigned to sets of users or sets of profiles.
`autoGreeting`	string	Specifies the greeting that displays when a customer begins a chat with an agent.
`capacity`	int	Specifies the maximum number of chats in which an agent can be engaged at a time.

Field Name	Field Type	Description
`criticalWaitTime`	int	Specifies the number of seconds an agent can wait to answer an engaged chat before the chat tab flashes to alert the agent to answer it.
`enableAgentSneakPeek`	boolean	Specifies whether a supervisor can see the content of an agent's message before they send it to a customer (`true`) or not (`false`).
`enableAssistanceFlag`	boolean	Indicates whether agents can raise an assistance flag to notify a supervisor that they need help. Available in API version 35.0 and later.
`enableAutoAwayOnDecline`	boolean	Indicates whether an agent appears as "away" (`true`) or not (`false`) when an agent declines a chat with a customer.
`enableAutoAwayOnPushTimeout`	boolean	Indicates whether an agent appears as "away" (`true`) or not (`false`) when a chat request that's been pushed to the agent times out. Available in API version 34.0 and later.
`enableAgentFileTransfer`	boolean	Indicates whether file transfer is enabled for agents (`true`) or not (`false`). Available in API version 31.0 and later.
`enableChatConferencing`	boolean	Indicates whether chat conferencing is enabled for agents (`true`) or not (`false`). Available in API version 34.0 and later.
`enableChatTransfer`	boolean	Indicates whether chat transfer is enabled for agents (`true`) or not (`false`). Available in API version 31.0 and later.
`enableLogoutSound`	boolean	Indicates whether a sound will play (`true`) or not (`false`) when an agent logs out of Live Agent.
`enableNotifications`	boolean	Indicates whether notifications of incoming chats appear for agents (`true`) or not (`false`).
`enableRequestSound`	boolean	Indicates whether a sound will play (`true`) or not (`false`) when a customer requests to chat with an agent.
`enableSneakPeek`	boolean	Indicates whether previews of customers' messages are displayed as customers type (`true`) or not (`false`) in the agent's Live Agent window. Available in API version 29.0 and later.
`enableVisitorBlocking`	boolean	Indicates whether an agent can block a visitor by IP address (`true`) or not (`false`). Available in API version 34.0 and later.
`label`	string	Specifies the name of the configuration for agents' default chat settings.
`supervisorDefaultAgentStatusFilter`	SupervisorAgentStatusFilter (enumeration of type string)	Specifies the Live Agent status for filtering the Agent Status list in the Supervisor Panel. Valid values are: • `Online`

Field Name	Field Type	Description
		• `Away` • `Offline` Available in API version 29.0 and later.
`supervisorDefaultButtonFilter`	string	Specifies the default button for filtering the Agent Status list in the Supervisor Panel. Available in API version 29.0 and later.
`supervisorDefaultSkillFilter`	string	Specifies the default skill for filtering the Agent Status list in the Supervisor Panel. Available in API version 29.0 and later.
`supervisorSkills`	SupervisorAgentConfigSkills	Specifies the list of agent skills that are assigned to a supervisor, as specified in their assigned Live Agent configuration. Available in API version 29.0 and later.
`transferableButtons`	AgentConfigButtons	Specifies the list of chat buttons that agents can transfer chats to. Available in API version 31.0 and later.
`transferableSkills`	AgentConfigSkills	Specifies the list of skill groups that agents can transfer chats to. Available in API version 31.0 and later.

AgentConfigAssignments

Represents the assignments of an organization's profiles and users to a Live Agent configuration.

Fields

Field Name	Field Type	Description
`profiles`	AgentConfigProfileAssignments	Specifies the profiles that are associated with a specific agent configuration.
`users`	AgentConfigUserAssignments	Specifies the users that are associated with a specific agent configuration.

AgentConfigButtons

Represents the chat buttons that agents who are associated with the Live Agent configuration can transfer chats to.

Fields

Field Name	Field Type	Description
`button`	string[]	Specifies the chat buttons that agents can transfer chats to.

AgentConfigProfileAssignments

Represents the profiles associated with a specific Live Agent configuration.

Fields

Field Name	Field Type	Description
`profile`	string	Specifies the custom name of the profile associated with a specific agent configuration.

AgentConfigSkills

Represents the skill groups that agents who are associated with the Live Agent configuration can transfer chats to.

Fields

Field Name	Field Type	Description
`skill`	string[]	Specifies the skill groups that agents can transfer chats to.

AgentConfigUserAssignments

Represents the users associated with a specific Live Agent configuration.

Fields

Field Name	Field Type	Description
`user`	string	Specifies the username of the user associated with a specific agent configuration.

SupervisorAgentConfigSkills

Represents the agent skills associated with a supervisor's Live Agent configuration. Available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`skill`	string	Specifies the agent skills available for filtering the Agent Status list in the Supervisor Panel.

Declarative Metadata Sample Definition

This is a sample of a `liveChatAgentConfig` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveChatAgentConfig xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>My Agent Configuration 1</label>
    <autoGreeting>Hi, how can I help you?</autoGreeting>
    <capacity>5</capacity>
    <enableAutoAwayOnDecline>true</enableAutoAwayOnDecline>
    <enableLogoutSound>true</enableLogoutSound>
    <enableNotifications>true</enableNotifications>
    <enableRequestSound>true</enableRequestSound>
    <enableSneakPeek>true</enableSneakPeek>
    <assignments>
        <profiles>
            <profile>standard</profile>
        </profiles>
        <users>
            <user>jdoe@acme.com</user>
        </users>
    </assignments>
</LiveChatAgentConfig>
```

LiveChatButton

Represents a Live Agent deployment's settings for the button that customers click to chat with an agent and the chat window, such as the label that appears on the button and the pre-chat form that appears before a live chat begins. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveChatButton configurations are stored in the `<developer_name>.liveChatButton` file in the `liveChatButtons` directory.

Version

LiveChatButton is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`chasitorIdleTimeout`	int	Specifies the amount of idle time before the chat times out. The idle time starts being counted after the agent sends the last chat message. Available in API version 35.0 and later.

Field Name	Field Type	Description
chasitorIdleTimeoutWarning	int	Specifies the amount of idle time before a warning appears. The idle time starts being counted after the agent sends the last chat message. Available in API version 35.0 and later.
chatPage	string	Specifies the page that hosts your chat if that page differs from the Live Agent chat window.
enableQueue	boolean	Indicates whether queuing is enabled (`true`) or not (`false`).
label	string	Specifies the text that appears on the button.
numberOfReroutingAttempts	int	Specifies the number of times a chat request can be rerouted to available agents if all agents reject the chat request. Available in API version 30.0 and later.
offlineImage	string	Specifies the image that appears on the button when no agents are available to chat.
onlineImage	string	Specifies the image that appears on the button when agents are available to chat.
optionsCustomRoutingIsEnabled	boolean	Indicates whether custom routing is enabled for incoming chat requests (`true`) or not (`false`). Available in API version 30.0 and later.
optionsHasChasitorIdleTimeout	boolean	Indicates whether the visitor idle timeout feature is enabled. Available in API version 35.0 and later.
optionsHasInviteAfterAccept	boolean	Indicates whether a new chat invitation triggers after a customer accepts a previous chat invitation (`true`) or not (`false`).
optionsHasInviteAfterReject	boolean	Indicates whether a new chat invitation triggers after a customer rejects a previous chat invitation (`true`) or not (`false`).
optionsHasRerouteDeclinedRequest	boolean	Indicates whether a chat request, which has been rejected by all available agents, should be rerouted to available agents again (`true`) or not (`false`). Available in API version 30.0 and later.
optionsIsAutoAccept	boolean	Indicates whether a chat request should be automatically accepted by the agent it's assigned to (`true`) or not (`false`). For chat buttons and automated chat invitations with `routingType` set to `MostAvailable` or `LeastActive`. Available in API version 30.0 and later.

Field Name	Field Type	Description
`optionsIsInviteAutoRemove`	boolean	Indicates whether a chat invitation is set to automatically disappear from a customer's screen after a certain amount of time (`true`) or not (`false`).
`overallQueueLength`	int	Specifies the maximum number of chat requests that are allowed to queue.
`perAgentQueueLength`	int	Specifies the number of chat requests that are allowed to queue for an agent with the required skills.
`postChatPage`	string	Specifies the name of the post-chat form to which customers are routed when the chat ends.
`postChatUrl`	string	Specifies the URL of the post-chat form to which customers are routed when the chat ends.
`preChatFormPage`	string	Specifies the name of the pre-chat form to which customers are routed before a chat begins.
`preChatFormUrl`	string	Specifies the URL of the pre-chat form to which customers are routed when the chat begins.
`pushTimeOut`	int	Specifies the number of seconds an agent has to answer an incoming chat request before the request is routed to another agent.
`routingType`	LiveChatButtonRoutingType (enumeration of type string)	Specifies how incoming chats should be routed to agents when a customer pushes a button. Valid values are: • `Choice` • `LeastActive` • `MostAvailable`
`site`	string	Specifies the Force.com site that hosts your custom chat button images or custom chat page. Note: You must have the "CustomDomain" permission enabled in your organization before you can use a Force.com site with Live Agent.
`skills`	LiveChatButtonSkills	Specifies the skills associated with the button. When a customer clicks the button to chat, they are automatically routed to agents with those skills.
`windowLanguage`	Language	Specifies the language preferences for the chat window associated with the button.

LiveChatButtonSkills

Represents the skills associated with a chat button.

Fields

Field Name	Field Type	Description
`skill`	string	Specifies the name of the skill.

Declarative Metadata Sample Definition

This is a sample of a `liveChatButton` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveChatButton xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>My Button 1</label>
    <chatPage>ChatterAnswersLogin</chatPage>
    <enableQueue>true</enableQueue>
    <offlineImage>MyOfflineButton</offlineImage>
    <onlineImage>MyOnlineButton</onlineImage>
    <perAgentQueueLength>5</perAgentQueueLength>
    <postchatPage>AnswersHome</postchatPage>
    <prechatFormPage>AccountVF</prechatFormPage>
    <pushTimeOut>300</pushTimeOut>
    <routingType>LeastActive</routingType>
    <site>LiveAgentSite</site>
    <skills>
        <skill>Skill1</skill>
        <skill>Skill2</skill>
    </skills>
</LiveChatButton>
```

Note: If you update your chat button through the Metadata API, be sure to update all Web pages that use the same chat button code.

LiveChatDeployment

Represents the configuration settings for a specific Live Agent deployment, such as the branding image for the deployment and whether or not chat transcripts are automatically saved. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveChatDeployment values are stored in the `<developer_name>.liveChatDeployment` file in the `liveChatDeployments` directory.

Version

LiveChatDeployment is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`brandingImage`	string	Specifies the branding image for the deployment.
`displayQueuePosition`	boolean	(Pilot) Determines whether a customer's queue position is displayed in a standard chat window while the customer waits for an agent to respond to the chat request (`true`) or not (`false`). This field is available as a pilot in API version 32.0. To enable this field, contact Salesforce.
`domainWhiteList`	LiveChatDeploymentDomainWhiteList	Specifies the list of domains that can host the deployment.
`enablePrechatApi`	boolean	Indicates whether or not the pre-chat API is enabled for the deployment (`true`) or not (`false`).
`enableTranscriptSave`	boolean	Indicates whether chat transcripts are automatically saved after a chat ends (`true`) or not (`false`).
`label`	string	Specifies the name of the deployment.
`mobileBrandingImage`	string	Specifies the branding image for the deployment that appears when customers access the deployment on a mobile device.
`site`	string	Specifies the site that hosts the images for the deployment. Note: You must have the "CustomDomain" permission enabled in your organization before you can use a Force.com site with Live Agent.
`windowTitle`	string	Specifies the title of the window associated with the deployment.

LiveChatDeploymentDomainWhiteList

Represents a Live Agent deployment's domain whitelist.

Fields

Field Name	Field Type	Description
domain	string	Specifies a domain that can host the deployment.

Declarative Metadata Sample Definition

This is a sample of a `liveChatDeployment` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveChatDeployment xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>My Deployment 1</label>
    <brandingImage>pkb_image_bannerBg</brandingImage>
    <mobileBrandingImage>pkb_image_bgBottom</mobileBrandingImage>
    <domainWhiteList>
        <domain>mydomain</domain>
        <domain>test</domain>
    </domainWhiteList>
    <enableTranscriptSave>true</enableTranscriptSave>
    <site>GL_Knowledge_Base</site>
    <windowTitle>My window title</windowTitle>
</LiveChatDeployment>
```

Note: If you update your deployment through the Metadata API, be sure to update all Web pages that use the same deployment code.

LiveChatSensitiveDataRule

Represents a rule for masking or deleting data of a specified pattern. Written as a regular expression (regex).

Use this object to mask or delete data of specified patterns, such as credit card, social security, phone and account numbers, or even profanity. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveChatSensitiveDataRule components have the suffix `.liveChatSensitiveDataRule` and are stored in the `liveChatSensitiveDataRule` folder.

Version

LiveChatSensitiveDataRule components are available in API version 35.0 and later.

Fields

Field Name	Field Type	Description
`actionType`	SensitiveDataActionType (enumeration of type string)	Required. The action to take on the text when the sensitive data rule is triggered. Possbile values are: • `Remove` • `Replace`
`description`	string	The description of the sensitive data rule—for example, "Block social security numbers."
`enforceOn`	int	Required. Determines the roles on which the rule is enforced. The value is determined using bitwise OR operation. There are seven possible values: **1.** Rule enforced on Agent **2.** Rule enforced on Visitor **3.** Rule enforced on Agent and Visitor **4.** Rule enforced on Supervisor **5.** Rule enforced on Agent and Supervisor **6.** Rule enforced on Visitor and Supervisor **7.** Rule enforced on Agent, Visitor, and Supervisor
`isEnabled`	boolean	Required. Specifies whether a sensitive data rule is active (`true`) or not (`false`). Default value (if none is provided) is `false`.
`pattern`	string	Required. The pattern of text blocked by the rule. Written as a JavaScript regular expression (regex).
`replacement`	string	The string of characters that replaces the blocked text (if `ActionType` *`Replace`* is selected).

Declarative Metadata Sample Definition

The following is an example of a LiveChatSensitiveDataRule component.

```
<LiveChatSensitiveDataRule xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionType>REPLACE</actionType>
    <enforceOn>7</enforceOn>
    <isEnabled>true</isEnabled>
    <pattern>[aeiou]</pattern>
    <replacement>œ</replacement>
</LiveChatSensitiveDataRule>
```

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

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <!-- To be used from
```

```
support.liveagent.testsuite.unifiedouting.testDeployButtonMDAPIWithExistingQueue -->
    <apiAccessLevel>Unrestricted</apiAccessLevel>

    <types>
        <members>Change_For_all</members>
        <name>LiveChatSensitiveDataRule</name>
    </types>

    <version>35.0</version>
</Package>
```

ManagedTopics

Represents navigational and featured topics managed in a community. A specific community is represented by the Network component.

Note: The related network must exist before you deploy managed topics. (This occurs automatically when deploying an entire organization.)

File Suffix and Directory Location

Components have the suffix managedTopics and are stored in the managedTopics folder. In that folder, you'll find separate files for each network (for example, `NetworkNameA.managedTopics` and `NetworkNameB.managedTopics`).

Version

ManagedTopics components are available in API version 32.0 and later.

Fields

Field Name	Field Type	Description
`ManagedTopic`	ManagedTopic	Represents a specific navigational or featured topic.

ManagedTopic

Field Name	Field Type	Description
`name`	string	The topic name.
`managedTopicType`	string	The topic type: "Navigational" or "Featured"
`topicDescription`	string	An optional description of topic contents. This field is accessible only via the API; there is no corollary in the user interface.
`parentName`	string	The name of a parent topic for which this topic is a child. Child topics are accessible from the subtopics section of the parent topic page and their feeds are added to the parent topic feed. Only navigational topics support parent-child relationships.

Field Name	Field Type	Description
position	int	The placement of this topic relative to others of the same type. The results differ depending on topic type: • For top-level navigational topics, position arranges the Topics menu in the community. • For child navigational topics, it arranges sibling topics in the subtopics section. • For featured topics, it arranges topic thumbnail images on the community home page. Enter a number between 0 and 24. (The maximum amount of navigational or featured topics is 25.)

Declarative Metadata Sample Definition

The following example retrieves or deploys managed topics for all networks:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>ManagedTopics</name>
    </types>
    <version>32.0</version>
</Package>
```

The following example shows a package.xml file referencing the ManagedTopics component:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>NetworkName</members>
        <name>ManagedTopics</name>
    </types>
    <version>32.0</version>
</Package>
```

The following example shows the ManagedTopics component itself:

```
<?xml version="1.0" encoding="UTF-8"?>
<ManagedTopics>
    <ManagedTopic>
        <name>Running</name>
        <managedTopicType>Navigational</managedTopicType>
        <topicDescription>Training advice</topicDescription>
        <parentName></parentName>
        <position>0</position>
    </ManagedTopic>
    <ManagedTopic>
        <name>Hiking</name>
        <managedTopicType>Navigational</managedTopicType>
        <topicDescription>Routes and gear</topicDescription>
```

```
        <parentName></parentName>
        <position>1</position>
    </ManagedTopic>
        <ManagedTopic>
            <name>Trails</name>
            <managedTopicType>Navigational</managedTopicType>
            <topicDescription>Maps for local favorites</topicDescription>
            <parentName>Hiking</parentName>
            <position>0</position>
        </ManagedTopic>
        <ManagedTopic>
            <name>Backpacks</name>
            <managedTopicType>Navigational</managedTopicType>
            <topicDescription>Recommended models</topicDescription>
            <parentName>Hiking</parentName>
            <position>1</position>
        </ManagedTopic>
    <ManagedTopic>
        <name>Footwear</name>
        <managedTopicType>Featured</managedTopicType>
        <topicDescription>Suggested types for each sport</topicDescription>
        <parentName></parentName>
        <position>0</position>
    </ManagedTopic>
    <ManagedTopic>
        <name>Conditioning</name>
        <managedTopicType>Featured</managedTopicType>
        <topicDescription>How to get fit for any activity</topicDescription>
        <parentName></parentName>
        <position>1</position>
    </ManagedTopic>
</ManagedTopics>
```

MatchingRule

Represents a matching rule that is used to identify duplicate records. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Matching rule components have the `.matchingRule` suffix and are stored in the `matchingRules` folder. The name of the component file is the standard or custom object name that is associated with the matching rule.

Version

MatchingRule is available in API version 33.0 and later.

Fields

Field Name	Field Type	Description
booleanFilter	string	Specifies filter logic conditions. For more information on filter logic, see "Getting the Most Out of Filter Logic" in the Salesforce Help.
description	string	The description of the matching rule.
label	string	Required. The name of the matching rule.
matchingRuleItems	MatchingRuleItem	The criteria that make up a matching rule.
ruleStatus	MatchingRuleStatus (enumeration of type string)	Required. The activation status of the matching rule. Values are: • *Inactive* • *Deactivating* • *DeactivationFailed* • *Active* • *Activating* • *ActivationFailed* Important: The only valid values you can declare when deploying a package are *Active* and *Inactive*.

MatchingRuleItem

Field Name	Field Type	Description
blankValueBehavior	BlankValueBehavior (enumeration of type string)	Specifies how blank fields affect whether the fields being compared are considered matches. Valid values are: • *MatchBlanks* • *NullNotAllowed* (default)
fieldName	string	Required. Indicates which field to compare when determining if a record is similar enough to an existing record to be considered a match.
matchingMethod	MatchingMethod (enumeration of type string)	Required. Defines how the fields are compared. Choose between the exact matching method and various fuzzy matching methods. Valid values are: • *Exact* • *FirstName* • *LastName* • *CompanyName* • *Phone* • *City* • *Street* • *Zip*

Field Name	Field Type	Description
		• *Title* For details on each matching method, see "Matching Methods Used with Matching Rules" in the Salesforce Help.

Declarative Metadata Sample Definition

The following is a sample XML definition of a matching rule. A matching rule can be associated with either a standard or a custom object.

```
<?xml version="1.0" encoding="UTF-8"?>
<MatchingRules xmlns="http://soap.sforce.com/2006/04/metadata">
<matchingRules>
<fullName>AccountMatchingRule</fullName>
<label>Matching rule for accounts</label>
<description>this is sample rule description</description>
<matchingRuleItems>
<blankValueBehavior>NullNotAllowed</blankValueBehavior>
<fieldName>BillingCity</fieldName>
<matchingMethod>City</matchingMethod>
</matchingRuleItems>

<matchingRuleItems>
<blankValueBehavior>NullNotAllowed</blankValueBehavior>
<fieldName>Name</fieldName>
<matchingMethod>CompanyName</matchingMethod>
</matchingRuleItems>

<ruleStatus>Inactive</ruleStatus>
</matchingRules>
</MatchingRules>
```

The following `package.xml` shows how to reference a matching rule by name. It specifies the type name of MatchingRule.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
<types>
<members>Account.AccountMatchingRule</members>
<name>MatchingRule</name>
</types>
<version>36.0</version>
</Package>
```

The following `package.xml` shows how to reference all matching rules by specifying the plural MatchingRules type name and using a wildcard to include all members.

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

```
<version>36.0</version>
</Package>
```

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 Salesforce user interface.

Version

Metadata components are available in API version 10.0 and later.

Fields

Field Name	Field Type	Description
`fullName`	string	Required. The name of the component. If a field, the name must specify the parent object, for example `Account.FirstName`. The `__c` suffix must be appended to custom object names and custom field names when you are setting the `fullName`. For example, a custom field in a custom object could have a `fullName` of `MyCustomObject__c.MyCustomField__c`. To reference a component in a package, prepend the package's namespace prefix to the component name in the `fullName` field. Use the following syntax: ***`namespacePrefix__ComponentName`***. For example, for the custom field component `MyCustomObject__c.MyCustomField__c` and the namespace `MyNS`, the full name is `MyNS__MyCustomObject__c.MyCustomField__c`. Note: A namespace prefix is a 1 to 15-character alphanumeric identifier that distinguishes your package and its contents from other publishers' packages. For more information, see "Register a Namespace Prefix" in the Salesforce Help.

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	Required. The name of the component. The `fullName` 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. 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.

SEE ALSO:

Metadata

MilestoneType

Represents the name and description of a milestone, which you can use in an entitlement process to track important steps in cases. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Milestone types are stored in the `milestoneTypes` directory of the corresponding package directory. The extension is `.milestoneType`.

Version

MilestoneType is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`description`	string	The description of the milestone.
`RecurrenceType`	~~MilestoneRecurrenceType~~ (enumeration of type string)	The type of recurrence for the milestone. Available in API version 29.0 and later. Valid values are: • `none`—Specifies no recurrence for the milestone. The milestone occurs only once until the entitlement process exits. • `recursIndependently`—Specifies independent recurrence for the milestone. • `recursChained`—Specifies sequential recurrence for the milestone.

Declarative Metadata Sample Definition

This is a sample milestone type.

```
<?xml version="1.0" encoding="UTF-8"?>
<MilestoneType xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>First Response Time</description>
</MilestoneType>
```

And, here's the sample `package.xml` file that references the MilestoneType component definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>* or a valid name of a milestone type</members>
        <name>MilestoneType</name>
    </types>
    <version>29.0</version>
</Package>
```

ModerationRule

Represents a rule used in your community to moderate user-generated content. Each rule specifies the user-generated content the rule applies to, the criteria to enforce the rule on, and the moderation action to take. You can create rules that block, flag, or replace user-generated content that contains offensive language or inappropriate content. This type extends the Metadata metadata type and inherits its `fullName` field.

Community moderation rules created with the Metadata API are more powerful than moderation rules set up in the Community Management UI. For example, in the UI you could create a rule that moderates posts and comments. In the Metadata API you could create a rule that moderates only the Link Name of a Link feed type. Use the Metadata API to express complex moderation rules.

Important: Don't update moderation rules you create using the Metadata API in the Community Management UI. If you do, you overwrite relevant Metadata API fields or the fields are ignored.

Keep the following things in mind when creating moderation rules:

- Your organization can have up to 30 rules. This limit is per organization, not per community.
- Each rule can have up to three criteria.
- Rules that block content run first, followed by rules that replace content, then rules that flag content. If two or more rules perform the same action, the oldest rule runs first.

File Suffix and Directory Location

ModerationRule components have the suffix `.rule` and are stored in the `moderation` directory of the corresponding package directory. The file name format follows *`community_name.moderation_rule_developer_name`*`.rule`.

Version

ModerationRule components are available in API version 36.0 and later.

Special Access Rules

To view, create, edit, and delete moderation rules, you need the Manage Communities or Create and Set Up Communities permission.

Fields

Field Name	Field Type	Description
`action`	ModerationRuleAction	Required. Indicates the moderation action that you want to take.
`activate`	boolean	Required. Indicates whether the moderation rule is active (`true`) or inactive (`false`).
`description`	string	A description of the moderation rule.
`entitiesAndFields`	ModeratedEntityField[]	Indicates the types of user-generated content this moderation rule applies to.
`masterLabel`	string	Required. Label for the moderation rule.
`userMessage`	string	The message you want your community members to see when their content is blocked. Use the *`%BLOCKED_KEYWORD%`* variable to display up to five blocked words in the user message. If you don't specify a message, the user sees the standard message: "You can't use *`%BLOCKED_KEYWORD%`* or other inappropriate words in this community. Review your content and try again."

ModerationRuleAction

The moderation action to take when your criteria is matched.

Field Name	Field Type	Description
`ModerationRuleAction`	ModerationRuleAction (enumeration of type string)	Required. Indicates the moderation action to take when your criteria is matched. The valid values are: • `Block` • `Replace` • `Flag`

ModeratedEntityField

The fields and entities you want to moderate.

Field Name	Field Type	Description
`entityName`	string	Required. Indicates the types of user-generated content the moderation rule applies to. Post and comments only apply to content created in groups and user profiles. All feed types, such as polls and links, are supported.
`fieldName`	string	Indicates the field the moderation rule applies to. Note: To moderate feed posts, use `entityName FeedItem` with `fieldName RawBody`. To moderate feed comments, use `entityName FeedComment` with `fieldName RawCommentBody` These two field names are internal only and don't appear in other API documentation.
`keywordList`	KeywordList string	Indicates the keyword list that you want to moderate against.

Declarative Metadata Sample Definition

The following is an example of a ModerationRule component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ModerationRule xmlns="http://soap.sforce.com/2006/04/metadata">
   <description>Blocks Bad Word List in posts, comments, Link URLs, titles, and poll
choices.</description>
   <masterLabel>Blocking Rule</masterLabel>
   <action>Block</action>
   <active>true</active>
   <userMessage>You can't use %BLOCKED_KEYWORD% or other inappropriate words in this
community. Review your content and try again.</userMessage>
   <!-- Applies the rule to FeedComment.RawCommentBody (an internal only field), if it
contains words from the keyword list specified -->
   <entitiesAndFields>
     <entityName>FeedComment</entityName>
     <fieldName>RawCommentBody</fieldName>
     <keywordList>community1.badword_list</keywordList>
   </entitiesAndFields>
   <entitiesAndFields>
     <entityName>FeedItem</entityName>
```

```
    <fieldName>LinkUrl</fieldName>
    <keywordList>community1.badword_list</keywordList>
  </entitiesAndFields>
  <!-- Applies the rule to FeedItem.RawBody (an internal only field), if it contains words
 from the keyword list specified -->
  <entitiesAndFields>
    <entityName>FeedItem</entityName>
    <fieldName>RawBody</fieldName>
    <keywordList>community1.badword_list</keywordList>
  </entitiesAndFields>
  <entitiesAndFields>
    <entityName>FeedItem</entityName>
    <fieldName>Title</fieldName>
    <keywordList>community1.badword_list</keywordList>
  </entitiesAndFields>
  <entitiesAndFields>
    <entityName>FeedPollChoice</entityName>
    <fieldName>ChoiceBody</fieldName>
    <keywordList>community1.badword_list</keywordList>
  </entitiesAndFields>
</ModerationRule>
```

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>
    <name>ModerationRule</name>
    <members>community1.blocking_rule</members>
  </types>
  <version>36.0</version>
</Package>
```

NamedCredential

Represents a named credential, which specifies the URL of a callout endpoint and its required authentication parameters in one definition. A named credential can be specified as an endpoint to simplify the setup of authenticated callouts.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

NamedCredential components have the suffix `.namedCredential` and are stored in the `namedCredentials` folder.

Version

NamedCredential components are available in API version 33.0 and later.

Fields

Field Name	Field Type	Description
`authProvider`	string	The authentication provider that the AuthProvider component represents.
`certificate`	string	If you specify a certificate, your Salesforce org supplies it when establishing each two-way SSL connection with the external system. The certificate is used for digital signatures, which verify that requests are coming from your Salesforce org.
`endpoint`	string	The URL or root URL of the callout endpoint. Corresponds to `URL` in the user interface.
`label`	string	A user-friendly name for the named credential that appears in the Salesforce user interface, such as in list views.
`oauthRefreshToken`	string	The OAuth refresh token. Used to obtain a new access token for an end user when a token expires.
`oauthScope`	string	Specifies the scope of permissions to request for the access token. Corresponds to `Scope` in the user interface.
`oauthToken`	string	The access token that's issued by your authorization server.
`password`	string	The password to be used by your org to access the external system. Ensure that the credentials have adequate privileges to access the external system. Depending on how you set up access, you might need to provide the administrator password.
`principalType`	`External PrincipalType` (enumeration of type string)	Determines whether you're using one set or multiple sets of credentials to access the external system. Corresponds to `Identity Type` in the user interface. The valid values are: • `Anonymous` • `PerUser` • `NamedUser`
`protocol`	`Authentication Protocol` (enumeration of type string)	The authentication protocol for accessing the external system. The valid values are: • `NoAuthentication` • `Oauth` • `Password`
`username`	string	The username to be used by your org to access the external system. Ensure that the credentials have adequate privileges for performing callouts to the external system. Depending on how you set up access, you might need to provide the administrator username.

Declarative Metadata Sample Definition

The following is an example of a NamedCredential component.

```
<?xml version="1.0" encoding="UTF-8"?>
<NamedCredential xmlns="http://soap.sforce.com/2006/04/metadata">
<endpoint>https://my_endpoint.example.com</endpoint>
<label>My Named Credential</label>
<principalType>PerUser</principalType>
<protocol>NoAuthentication</protocol>
</NamedCredential>
```

Network

Represents a community. Communities are branded spaces for your employees, customers, and partners to connect. You can customize and create communities to meet your business needs, then transition seamlessly between them. Use the Network component for Salesforce Communities. If you want to create zones that contain Chatter Answers and Ideas, use the Community (Zone) component. This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

Network components are stored in the `networks` directory of the corresponding package directory. The file name matches the community name, and the extension is `.network`.

Version

This object is available in API version 28.0 and later.

Fields

Field	Field Type	Description
`allowedExtensions`	string	Specifies the types of files allowed in your community. This whitelist of file types lets you control what your community members upload and also prevents spammers from polluting your community with inappropriate files. Available in API version 36.0 and later.
`allowMembersToFlag`	boolean	Determines whether users in the community can flag posts or comments as inappropriate. Flagged items are sent to a community moderator for review. Available in API version 29.0 and later.
`branding`	Branding	The color scheme, header, and footer used in the community.
`caseCommentEmailTemplate`	string	Email template used when notifying community members when a case comment has been modified or added to a case.

Field	Field Type	Description
changePasswordTemplate	string	Email template used when notifying a user that their password has been reset.
description	string	Description of the community.
emailSenderAddress	string	Email address from which community emails are sent.
emailSenderName	string	Name from which community emails are sent.
enableGuestChatter	boolean	Specifies whether guest users can access public Chatter groups in the community without logging in.
enableInvitation	boolean	Determines whether users can invite others to the community.
enableKnowledgeable	boolean	Determines if community members can see who's knowledgeable on topics and endorse people for their knowledge on a topic. Available in API version 30.0 and later.
enableNicknameDisplay	boolean	Determines if user nicknames display instead of their first and last names in most places in the community. Set to `false` by default. Available in API version 32.0 and later.
enablePrivateMessages	boolean	Determines if community members can send and receive private messages. Available in API version 30.0 and later.
enableReputation	boolean	Determines if reputation is calculated and displayed for community members. Available in API version 31.0 and later. If enabled, `reputationLevels` and `reputationPointsRules` are used. If no `reputationLevels` or `reputationPointsRules` are not defined in the data file, the default values are used.
forgotPasswordTemplate	string	The email template used when a user forgets their password.
maxFileSizeKb	int	Specifies the maximum file size (in KBs) that members can upload in your community. Available in API version 36.0 and later. Enter a number between 3072 KB and your org's maximum file size. To use the default limit of 2 GB, leave this field empty.
networkMemberGroups	NetworkMemberGroups	The profiles and permission sets that have access to the community. Users with these profiles or permission sets are members of the community. Note: If a Chatter customer (from a customer group) is assigned a permission set that is also

Field	Field Type	Description
		associated with a community, the Chatter customer isn't added to the community.
`newSenderAddress`	string	Email address that has been entered as the new value for `EmailSenderAddress` but has not been verified yet. After a user has requested to change the sender email address and has successfully responded to the verification email, the `NewSenderAddress` value overwrites the value in `EmailSenderAddress`. This becomes the email address from which community emails are sent.
`picassoSite`	string	Name of the Site.com site linked to the community.
`reputationLevels`	ReputationLevelDefinitions	The reputation levels assigned to members when they accrue points by performing certain actions.
`reputationPointsRules`	ReputationPointsRules	The points members accrue when they perform certain defined actions.
`selfRegProfile`	string	The profile assigned to users who self register. This value is used only if `selfRegistration` is enabled for the community. Available in API version 29.0 and later.
`selfRegistration`	boolean	Determines whether self-registration is available for the community.
`sendWelcomeEmail`	boolean	Determines whether a welcome email is sent when a new user is added to the community.
`site`	string	The CustomSite associated with the community.
`status`	NetworkStatus[]	Status of the community. Available values are: • Live—The community is online and members can access it. • DownForMaintenance—The community was previously published, but was taken offline. Members with "Create and Set Up Communities" can still access the setup for offline communities regardless of profile or membership. Members are not able to access offline communities, but they still appear in the user interface drop-down as `CommunityName (Offline)`. • UnderConstruction—The community has not yet been published. Users with "Create and Set Up Communities" can access communities in this status if their profile is associated with the community. Once a community is published, it can never be in this status again.

Field	Field Type	Description
`tabs`	NetworkTabSet	The tabs that are available in the community. The user that created the community selected these tabs.
`urlPathPrefix`	string	The first part of the path on the site's URL that distinguishes this site from other sites. For example, if your site URL is `mycompany.force.com/partners`, then `partners` is the `urlPathPrefix`.
`welcomeTemplate`	string	The email template used when sending welcome emails to new community members.

Branding

Represents the branding and color scheme applied to the community.

Field	Field Type	Description
`loginFooterText`	string	The text that appears in the footer of the community login page.
`loginLogo`	string	The logo that appears on the community login page for external users.
`pageFooter`	string	An image that appears on the footer of the community pages. Must be an .html file.
`pageHeader`	string	An image that appears on the header of the community pages. Can be an .html, .gif, .jpg, or .png file.
`primaryColor`	string	The color used for the active tab.
`primaryComplementColor`	string	Font color used with `primaryColor`.
`quaternaryColor`	string	The background color for pages in the community.
`quaternaryComplementColor`	string	Font color used with `quaternaryColor`.
`secondaryColor`	string	The color used for the top borders of lists and tables.
`tertiaryColor`	string	The background color for section headers on edit and detail pages.
`tertiaryComplementColor`	string	Font color used with `tertiaryColor`.
`zeronaryColor`	string	The background color for the header.
`zeronaryComplementColor`	string	Font color used with `zeronaryColor`.

NetworkMemberGroup

Represents the profiles and permission sets that are assigned to the community. Users with one of the profiles or permission sets are members of the community, unless the user is a Chatter customer (from a customer group).

Field	Field Type	Description
`permissionSet`	string	A permission set that is assigned to the community. Note: If a Chatter customer (from a customer group) is assigned a permission set that is also associated with a community, the Chatter customer isn't added to the community.
`profile`	string	A profile that is part of the community.

ReputationBranding

Branding for the reputation level.

Field	Field Type	Description
`smallImage`	string	Custom image associated with a reputation level. Use files with these extensions: .jpeg, .png, or .gif. Images are stored as documents. If not specified, the default reputation level image is used. Available in API version 32.0 and later.

ReputationLevelDefinitions

Represents reputation levels members can achieve by performing certain defined actions in a community.

Field	Field Type	Description
`level`	ReputationLevel[]	Represents reputation levels.

ReputationLevel

Represents the name and lower value of the reputation level. The upper value is calculated by the application.

Field	Field Type	Description
`branding`	ReputationBranding[]	Represents any branding associated with the reputation level, specifically, the custom image for the reputation level. This field is optional. If not specified, the default reputation level image is used. Available in API version 32.0 and later.
`label`	string	Name of the reputation level. This field is optional. If not specified, one of the 10 defaults are used. • Level 1 • Level 2 • Level 3 • Level 4 • Level 5 • Level 6 • Level 7 • Level 8 • Level 9 • Level 10
`lowerThreshold`	double	Required. The lower value in the range for this reputation level. For example, if this reputation level is for points 1–50, 1 is the `lowerThreshold`.

ReputationPointsRules

Represents points rules in a community's point system.

Field	Field Type	Description
`pointsRule`	ReputationPointsRule[]	Represents events and their associated points.

ReputationPointsRule

Represents the event and associated point value for a points rule. When a user acts, they accrue the associated points.

Field	Field Type	Description
`eventType`	string	Required. The type of event a member has to perform to get points. The available values are: • `FeedItemWriteAPost` • `FeedItemWriteAComment` • `FeedItemReceiveAComment` • `FeedItemLikeSomething` • `FeedItemReceiveALike` • `FeedItemMentionSomeone` • `FeedItemSomeoneMentionsYou` • `FeedItemShareAPost` • `FeedItemSomeoneSharesYourPost` • `FeedItemPostAQuestion` • `FeedItemAnswerAQuestion` • `FeedItemReceiveAnAnswer` • `FeedItemMarkAnswerAsBest` • `FeedItemYourAnswerMarkedBest` • `FeedItemEndorseSomeoneForKnowledgeOnATopic` • `FeedItemEndorsedForKnowledgeOnATopic`
`points`	int	Required. The number of points a member gets for performing the event. The default number of points per event is: • FeedItemWriteAPost +1 • FeedItemWriteAComment: +1 • FeedItemReceiveAComment: +5 • FeedItemLikeSomething: +1 • FeedItemReceiveALike: +5 • FeedItemMentionSomeone: +1 • FeedItemSomeoneMentionsYou: +5 • FeedItemShareAPost: +1 • FeedItemSomeoneSharesYourPost: +5 • FeedItemPostAQuestion: +1 • FeedItemAnswerAQuestion: +5 • FeedItemReceiveAnAnswer: +5 • FeedItemMarkAnswerAsBest: +5 • FeedItemYourAnswerMarkedBest: +20 • FeedItemEndorseSomeoneForKnowledgeOnATopic: +5 • FeedItemEndorsedForKnowledgeOnATopic: +20

NetworkTabSet

Field	Field Type	Description
customTab	string	Custom tab that is part of the community.
defaultTab	string	The Home tab for the community. When members log in, this tab is the first page they see.
standardTab	string	Standard tab that is part of the community.

Declarative Metadata Sample Definition

A sample XML definition of a network.

```
<?xml version="1.0" encoding="UTF-8"?>
<Network xmlns="http://soap.sforce.com/2006/04/metadata">
    <allowMembersToFlag>true</allowMembersToFlag>
    <branding>
        <loginFooterText>salesforce.com</loginFooterText>
        <loginLogo>Communities_Shared_Document_Folder/header2_png.png</loginLogo>
        <pageFooter>Branding/footer_html.html</pageFooter>
        <pageHeader>Branding/header_Image.jpg</pageHeader>
        <primaryColor>#AF5800</primaryColor>
        <primaryComplementColor>#FFFFFF</primaryComplementColor>
        <quaternaryColor>#286FB8</quaternaryColor>
        <quaternaryComplementColor>#FFFFFF</quaternaryComplementColor>
        <secondaryColor>#000000</secondaryColor>
        <tertiaryColor>#FFFFFF</tertiaryColor>
        <tertiaryComplementColor>#222222</tertiaryComplementColor>
        <zeronaryColor>#0A3764</zeronaryColor>
        <zeronaryComplementColor>#FFFFFF</zeronaryComplementColor>
    </branding>

<changePasswordTemplate>unfiled$public/CommunityChangePasswordEmailTemplate</changePasswordTemplate>

    <description>Metadata Community</description>
    <emailSenderAddress>admin@networkMetadata.com</emailSenderAddress>
    <emailSenderName>Admin User</emailSenderName>
    <enableInvitation>false</enableInvitation>
    <enableKnowledgeable>true</enableKnowledgeable>
    <enableNicknameDisplay>false</enableNicknameDisplay>
    <enablePrivateMessages>true</enablePrivateMessages>
    <enableReputation>true</enableReputation>

<forgotPasswordTemplate>unfiled$public/CommunityForgotPasswordEmailTemplate</forgotPasswordTemplate>

    <networkMemberGroups>
        <permissionSet>Admin</permissionSet>
        <permissionSet>Standard</permissionSet>
        <permissionSet>ReadOnly</permissionSet>
        <profile>Admin</profile>
```

```
        <profile>Standard</profile>
        <profile>ReadOnly</profile>
    </networkMemberGroups>
    <reputationLevels>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_beginner.png</smallImage>
            </branding>
            <label>Beginner</label>
            <lowerThreshold>0</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_apprentice.png</smallImage>
            </branding>
            <label>Apprentice</label>
            <lowerThreshold>51</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_gettingthere.png</smallImage>
            </branding>
            <label>Getting There</label>
            <lowerThreshold>101</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_skilled.png</smallImage>
            </branding>
            <label>Skilled</label>
            <lowerThreshold>151</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_expert.png</smallImage>
            </branding>
            <label>Expert</label>
            <lowerThreshold>201</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_mentor.png</smallImage>
            </branding>
            <label>Mentor</label>
            <lowerThreshold>251</lowerThreshold>
        </level>
        <level>
            <branding>
```

```
                <smallImage>communities_shared
_document_folder/replevel_guru.png</smallImage>
            </branding>
            <label>Guru</label>
            <lowerThreshold>301</lowerThreshold>
        </level>
    </reputationLevels>
    <reputationPointsRules>
        <pointsRule>
            <eventType>FeedItemWriteAPost</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemWriteAComment</eventType>
            <points>3</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemReceiveAComment</eventType>
            <points>10</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemLikeSomething</eventType>
            <points>3</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemReceiveALike</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemMentionSomeone</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemSomeoneMentionsYou</eventType>
            <points>10</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemShareAPost</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemSomeoneSharesYourPost</eventType>
            <points>10</points>
        </pointsRule>
    </reputationPointsRules>
    <selfRegistration>false</selfRegistration>
    <sendWelcomeEmail>true</sendWelcomeEmail>
    <site>Network_11</site>
    <status>UnderConstruction</status>
    <tabs>
        <defaultTab>Chatter</defaultTab>
        <standardTab>Chatter</standardTab>
        <standardTab>Account</standardTab>
        <standardTab>Campaign</standardTab>
```

```
        <standardTab>Case</standardTab>
        <standardTab>Console</standardTab>
        <standardTab>Contact</standardTab>
        <standardTab>Contract</standardTab>
        <standardTab>Dashboard</standardTab>
        <standardTab>JigsawSearch</standardTab>
        <standardTab>File</standardTab>
        <standardTab>CollaborationGroup</standardTab>
        <standardTab>home</standardTab>
        <standardTab>Idea</standardTab>
        <standardTab>Lead</standardTab>
        <standardTab>Opportunity</standardTab>
        <standardTab>Product2</standardTab>
        <standardTab>UserProfile</standardTab>
        <standardTab>report</standardTab>
        <standardTab>Solution</standardTab>
    </tabs>
    <urlPathPrefix>network1</urlPathPrefix>
    <welcomeTemplate>unfiled$public/CommunityWelcomeEmailTemplate</welcomeTemplate>
</Network>
```

SEE ALSO:

Community (Zone)

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)	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: • 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. For more information, see "About API and Dynamic Apex Access in Packages" in the Salesforce online help.
`description`	string	A short description of the package.

Name	Type	Description
`fullName`	string	The package name used as a 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 end with an underscore, and 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 metadata components of the type specified in the `<name>` element. To retrieve a standard object, specify it by name. For example `<members>Account</members>` will retrieve the standard Account object.
`name`	string	The type of metadata component to be retrieved. For example `<name>CustomObject</name>` will retrieve one or more custom objects as specified in the `<members>` element.

SEE ALSO:

Sample package.xml Manifest Files

PathAssistant

Represents Sales Path records for Opportunity.This type extends the Metadata metadata type and inherits its `fullName` field.

Note the following when working with PathAssistant:

- Only one sales path can be created per record type, including __Master__ record type.

- Rich text guidance information cannot be retrieved or deployed from or to translation workbench.
- The preference does not need to be on to retrieve or deploy PathAssistant.

File Suffix and Directory Location

PathAssistant components have the suffix `.pathAssistant` and are stored in the `pathAssistants` folder.

Version

PathAssistant components are available in API version 34.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Indicates whether the sales path is active (`true`) or not (`false`).
`entityName`	string	Required. The entity name. This is hard coded for Opportunity. This field is not updateable.
`fieldName`	string	Required. The field name. This is hard coded for StageName. This field is not updateable.
`masterLabel`	string	Required. The master label of the sales path.
`pathAssistantSteps`	PathAssistantStep[] on page 432	List of all the steps that have been configured with fields and guidance information. Note that a missing step in the .xml file means it has not been configured, not that it doesn't exist.
`recordTypeName`	string	Required. The name of the record type associated with the sales path. This field is not updateable.

PathAssistantStep

Represents the steps or stages in Sales Path.

Field Name	Field Type	Description
`fieldNames`	string	All the fields in `entityName` that will display in this step.
`info`	string	The guidance information displayed in this step.
`picklistValueName`	string	Required. The stage picklist value associated with the step.

Declarative Metadata Sample Definition

The following is an example of a PathAssistant component.

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

```
    <active>true</active>
    <entityName>Opportunity</entityName>
    <fieldName>StageName</fieldName>
    <masterLabel>Test Path</masterLabel>
    <pathAssistantSteps>
        <fieldNames>Amount</fieldNames>
        <fieldNames>CloseDate</fieldNames>
        <info>Some Text</info>
        <picklistValueName>Id. Decision Makers</picklistValueName>
    </pathAssistantSteps>
    <pathAssistantSteps>
        <fieldNames>Amount</fieldNames>
        <fieldNames>CloseDate</fieldNames>
        <info>Some Text</info>
        <picklistValueName>Proposal/Price Quote</picklistValueName>
    </pathAssistantSteps>
    <recordTypeName>Test_Record_Type</recordTypeName>
</PathAssistant>
```

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>Opportunity.Test_Busines_Process</members>
        <name>BusinessProcess</name>
    </types>
    <types>
        <members>Opportunity.StageName</members>
        <members>Lead.LeadSource</members>
        <members>Opportunity.Type</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>Test_Path</members>
        <name>PathAssistant</name>
    </types>
    <types>
        <members>Opportunity.Test_Record_Type</members>
        <name>RecordType</name>
    </types>
    <types>
        <members>PathAssistant</members>
        <name>Settings</name>
    </types>
    <version>36.0</version>
</Package>
```

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. This type 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
- Apps
- Custom field permissions
- Custom object permissions
- Custom tab settings
- External data sources
- Record types
- Visualforce pages

For more information, see Managed Component Access in Sample `package.xml` Manifest Files on page 20.

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_Perms* is stored in `permissionsets/User_Management_Perms.permissionset`.

Version

Permission sets are available in API version 22.0 and later.

Fields

Field	Field Type	Description
`applicationVisibilities`	PermissionSetApplicationVisibility[]	Indicates which apps are visible to users assigned to this permission set. Available in API version 29.0 and later. In API version 29.0, this field supports custom apps only. In API version 30.0 and later, this field supports both standard and custom apps.
`classAccesses`	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.
`customPermissions`	PermissionSetCustomPermissions[]	Indicates which custom permissions are available to users assigned to this permission set. Available in API version 31.0 and later.
`description`	string	The permission set description. Limit: 255 characters.
`externalDataSourceAccesses`	PermissionSetExternal DataSourceAccess[]	Indicates which data sources with identity type of `Per User` are available to users assigned to this permission set. Available in API version 27.0 and later.
`fieldPermissions`	PermissionSetFieldPermissions[]	Indicates which fields are accessible to a user assigned to this permission set, and the kind of access available

Field	Field Type	Description
		(readable or editable). Available in API version 23.0 and later.
`label`	string	The permission set label. Limit: 80 characters.
`objectPermissions`	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.
`pageAccesses`	PermissionSetApexPageAccess[]	Indicates which Visualforce pages that users assigned to this permission set can execute. Available in API version 23.0 and later.
`recordTypeVisibilities`	PermissionSetRecordTypeVisibility[]	Indicates which record types are visible to users assigned to this permission set. Available in API version 29.0 and later. This field is never retrieved or deployed for inactive record types.
`tabSettings`	PermissionSetTabSetting[]	Indicates the tab visibility settings for this permission set. Available in API version 26.0 and later.
`userLicense`	string	The `User License` for the permission set. A user license determines the baseline of features that the user can access. Every user must have exactly one user license.
`userPermissions`	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.

PermissionSetApplicationVisibility

PermissionSetApplicationVisibility determines whether an app is visible to a user assigned to this permission set.

Field Name	Field Type	Description
`application`	string	Required. The app name.
`visible`	boolean	Required. Indicates whether this app is visible to users assigned to this permission set (`true`) or not (`false`).

PermissionSetApexClassAccess

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

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

PermissionSetCustomPermissions

PermissionSetCustomPermissions represents the custom permissions access for users assigned to a permission set. Only enabled custom permissions are retrieved.

Field Name	Field Type	Description
`enabled`	boolean	Required. Indicates whether the custom permission is enabled (`true`) or not (`false`).
`name`	string	Required. The custom permission name.

PermissionSetExternalDataSourceAccess

PermissionSetExternalDataSourceAccess represents the data source access for users with identity type of `Per User`. Available in API version 27.0 and later.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the data source is enabled (`true`) or not (`false`).
`externalDataSource`	string	The name of the external data source.

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 (`true`) or not (`false`).
`field`	string	Required. The API name of the field (such as `Warehouse__c.Description__c`).
`readable`	boolean	Indicates whether the field can be read by the users assigned to this permission set (`true`) or not (`false`).

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 `object` field can be created by the users assigned to this permission set (`true`) or not (`false`).
`allowDelete`	boolean	Required. Indicates whether the object referenced by the `object` field can be deleted by the users assigned to this permission set (`true`) or not (`false`).
`allowEdit`	boolean	Required. Indicates whether the object referenced by the `object` field can be edited by the users assigned to this permission set (`true`) or not (`false`).
`allowRead`	boolean	Required. Indicates whether the object referenced by the `object` field can be viewed by the users assigned to this permission set (`true`) or not (`false`).
`modifyAllRecords`	boolean	Required. Indicates whether the object referenced by the `object` field can be viewed, edited, or deleted by the users assigned to this permission set (`true`) or not (`false`), 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 `Warehouse__c`).
`viewAllRecords`	boolean	Required. Indicates whether the object referenced by the `object` field can be viewed by the users assigned to this permission set (`true`) or not (`false`), 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.

PermissionSetApexPageAccess

PermissionSetApexPageAccess represents the Visualforce page access for users assigned to a permission set.

Field	Field Type	Description
`apexPage`	string	Required. The Visualforce page name.
`enabled`	boolean	Required. Indicates whether users assigned to this permission set can execute the Visualforce page (`true`) or not (`false`).

PermissionSetRecordTypeVisibility

PermissionSetRecordTypeVisibility represents the visibility of record types for this permission set.

Field	Field Type	Description
`recordType`	string	Required. The record type name, for example `Account.MyRecordType`.
`visible`	boolean	Required. Indicates whether the record type is visible to users assigned to this permission set (`true`) or not (`false`).

PermissionSetTabSetting

PermissionSetTabSetting represents the tab settings for a permission set.

Field	Field Type	Description
`tab`	string	Required. The tab name.
`visibility`	PermissionSetTabVisibility (enumeration of type string)	Required. Indicates the visibility settings for the tab. Valid values are: • `Available`—The tab is available on the All Tabs page. Individual users can customize their display to make the tab visible in any app. • `None`—The tab isn't available on the All Tabs page or visible in any apps. • `Visible`—The tab is available on the All Tabs page and appears in the visible tabs for its associated app. Individual users can customize their display to hide the tab or make it visible in other apps.

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 (`true`) or disabled (`false`).
`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>
        <members>Job_Request__c</members>
        <name>CustomObject</name>
    </types>
    <types>
        <members>JobApps__Recruiting</members>
        <name>CustomApplication</name>
    </types>
    <types>
        <members>Recruiting.DevManager</members>
        <name>RecordType</name>
    </types>
    <types>
        <members>*</members>
        <name>PermissionSet</name>
    </types>
    <version>36.0</version>
</Package>
```

PlatformCachePartition

Represents a partition in the Platform Cache. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

PlatformCachePartition components have the suffix `.cachePartition` and are stored in the `cachePartitions` folder.

Version

PlatformCachePartition components are available in API version 35.0 and later.

Special Access Rules

The "Author Apex" permission is required to deploy and retrieve PlatformCachePartition components.

Fields

Field Name	Field Type	Description
description	string	Describes the cache partition.
isDefaultPartition	boolean	Required. Indicates whether this cache partition is the default partition in your organization (true) or not (false).
masterLabel	string	Required. The label of the cache partition that appears in the Salesforce user interface.
platformCachePartitionTypes	PlatformCachePartitionType[]	An array of cache types that the partition can store.

PlatformCachePartitionType

Contains information about a partition type, including its minimum and allocated capacity.

Field Name	Field Type	Description
allocatedCapacity	int	Required. The total storage capacity, in MB, that is allocated for the cache type, including free, purchased, and trial cache. Purchased capacity includes organization-wide cache, which can be used in any partition, and namespace-specific cache, which can be used only in partitions associated with a namespace.
allocatedPurchasedCapacity	int	Required. The amount of namespace-specific purchased storage capacity, in MB, that is allocated for the cache type.
allocatedTrialCapacity	int	Required. The amount of trial cache space, in MB, that is allocated for the cache type.
cacheType	PlatformCacheType (enumeration of type string)	The type of cache. Valid values are: • Session—Session cache • Organization—Org cache

Declarative Metadata Sample Definition

The following is an example of a PlatformCachePartition component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformCachePartition xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Custom partition and marked as default.</description>
    <isDefaultPartition>true</isDefaultPartition>
    <masterLabel>myPartition</masterLabel>
    <platformCachePartitionTypes>
        <allocatedCapacity>10</allocatedCapacity>
        <allocatedPurchasedCapacity>5</allocatedPurchasedCapacity>
        <cacheType>Session</cacheType>
    </platformCachePartitionTypes>
```

```
    <platformCachePartitionTypes>
        <allocatedCapacity>5</allocatedCapacity>
        <allocatedPurchasedCapacity>5</allocatedPurchasedCapacity>
        <cacheType>Organization</cacheType>
    </platformCachePartitionTypes>
</PlatformCachePartition>
```

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>myPartition</members>
        <name>PlatformCachePartition</name>
    </types>
    <version>36.0</version>
</Package>
```

If a namespace is defined in your organization, add the namespace prefix to your partition name. For example:

```
<members>Namespace.myPartition</members>
```

To retrieve all cache partitions from your organization, use the wildcard character (*) as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>PlatformCachePartition</name>
    </types>
    <version>36.0</version>
</Package>
```

Portal

The Portal metadata type represents a partner portal or Customer Portal. It extends Metadata and inherits its `fullName` field. To use this metadata type, you must have a partner portal or Customer Portal enabled for your organization. For more information, see "Partner Portal Overview" and "Enabling Your Customer Portal" in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Force.com Portal components are stored in the `portals` directory of the corresponding package directory. The file name matches the portal name, and the extension is `.portal`.

Version

Force.com Portal components are available in API version 15.0 and later.

Fields

Field	Field Type	Description
`active`	boolean	Required. Denotes whether this portal is active.
`admin`	string	The full name of the user designated to administer the portal.
`defaultLanguage`	string	The default language for HTML messages for the portal. Use the abbreviation for the language, for example, en_US for United States English.
`description`	string	The portal description.
`emailSenderAddress`	string	Required. The email address used when sending emails using templates configured from the portal (for example, for resetting the password).
`emailSenderName`	string	Required. The name to display when sending emails using templates configured from the portal (for example, for resetting the password).
`enableSelfCloseCase`	boolean	For the Customer Portal, allows portal users to close their own cases.
`footerDocument`	string	The file to be used as the footer for this portal.
`forgotPassTemplate`	string	The email template to use when a user clicks the **Forgot Password** link.
`fullName`	string	Required. The name of the portal. 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.
`headerDocument`	string	The file to be used as the header for this portal.
`isSelfRegistrationActivated`	boolean	Determines whether self-registration is active or not for this portal.
`loginHeaderDocument`	string	The file to be used as the header for this portal's login page.
`logoDocument`	string	The file to be used as the logo for this portal.
`logoutUrl`	string	The URL that the user should be redirected to on logout.
`newCommentTemplate`	string	The email template to be used for auto-notifications on new case comments.
`newPassTemplate`	string	The email template to be used for auto-notifications on password reset.
`newUserTemplate`	string	The email template to be used for auto-notifications on new user creation.

Field	Field Type	Description
ownerNotifyTemplate	string	The email template to be used for auto-notifications on owner change.
selfRegNewUserUrl	string	The URL of the self-registration page.
selfRegUserDefaultProfile	string	The default profile for self-registered users.
selfRegUserDefaultRole	PortalRoles (enumeration of type string)	The default role for self-registered users. The valid values are: • Executive • Manager • User • PersonAccount
selfRegUserTemplate	string	The email template to be used for auto-notifications on self-registration.
showActionConfirmation	boolean	Determines whether or not confirmation messages are displayed for actions in the portal.
stylesheetDocument	string	The Document object to be used as the CSS stylesheet for this portal.
type	PortalType (enumeration of type string)	Required. The type for this portal. The valid values are: • CustomerSuccess • Partner

Declarative Metadata Sample Definition

A sample XML definition of a portal is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Portal xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <description>Customer Portal</description>
    <emailSenderName>rguest@albany.com</emailSenderName>
    <enableSelfCloseCase>false</enableSelfCloseCase>
    <forgotPassTemplate>unfiled$public/ChangePwdEmail</forgotPassTemplate>
    <isSelfRegistrationActivated>false</isSelfRegistrationActivated>
    <newPassTemplate>unfiled$public/ChangePwdEmail</newPassTemplate>
    <newUserTemplate>unfiled$public/NewUserEmail</newUserTemplate>
    <selfRegUserTemplate>unfiled$public/SelfRegUserEmail</selfRegUserTemplate>
    <showActionConfirmation>false</showActionConfirmation>
    <type>CustomerSuccess</type>
</Portal>
```

SEE ALSO:

CustomSite

PostTemplate

Represents the metadata associated with an approval post template for Approvals in Chatter. With approval post templates, you can customize the information included in approval request posts that appear in Chatter feeds. This type extends the Metadata metadata type and inherits its `fullName` field.

Note:

- You can only create an approval post template for an object that is both enabled for approvals and supported for Chatter feed tracking. You can't associate a post template to an approval process unless feed tracking for the object has been turned on.
- Deleting a custom field removes it from any approval post template that references it. Existing posts aren't affected. Undeleting the custom field restores it to the available fields list, but doesn't restore it to any approval post templates that previously contained it.
- Deleting (or undeleting) a custom object also deletes (or undeletes) its associated approval post templates and any of its approval request posts that are already in Chatter feeds.
- If you rename a custom object, approval post templates associated with it update accordingly.
- You can create custom approval post templates for one or more approval processes, but you can only associate custom post templates with approval processes after enabling Approvals in Chatter.

File Suffix and Directory Location

PostTemplate components have the suffix `.postTemplate` and are stored in the `postTemplates` folder.

Version

PostTemplate components are available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`default`	boolean	Required. Specifies whether this is the default post template for the given object. When set to `true`, this post template is used by approval processes that are associated with the same object and don't specify a post template. When an object has no default post template, each of its approval processes uses the system default post template, unless the approval process specifies its own post template.
`description`	string	Optional description of the post template.
`fields`	string[]	Required. An array of up to four fields to include in approval request posts. If you make your approval object a detail object in a master-detail relationship, the `Owner` field isn't available for approval page layouts or approval post templates.
`label`	string	Required. Name of the post template. This non-unique label is different from the unique name of the post template.

Declarative Metadata Sample Definition

The following is an example of a PostTemplate component:

```
<PostTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
    <default>false</default>
    <fields>NumberOfEmployees</fields>
    <fields>NumberofLocations__c</fields>
    <fields>PartnerAccount</fields>
    <fields>LeadCustomFieldNumber__c</fields>
    <label>My Lead Post Template</label>
</PostTemplate>
```

The following is an example package manifest that references the previous PostTemplate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Lead.leadtemplate</members>
        <name>PostTemplate</name>
    </types>
    <version>29.0</version>
</Package>
```

Profile

Represents a user profile. A profile defines a user's permission to perform different functions within Salesforce. This type 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
- Apps
- Custom field permissions
- Custom object permissions
- Custom tab settings
- External data sources
- Record types
- Visualforce pages

For more information, see Managed Component Access in Sample `package.xml` Manifest Files on page 20.

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
`applicationVisibilities`	ProfileApplicationVisibility[]	Indicates which apps are visible to users assigned to this profile. In API version 29.0 and earlier, this field supports custom apps only. In API version 30.0 and later, this field supports both standard and custom apps.
`classAccesses`	ProfileApexClassAccess[]	Indicates which top-level Apex classes have methods that users assigned to this profile can execute.
`custom`	boolean	Indicates whether the profile is a custom (`true`) or standard (`false`) profile. Available in API version 30.0 and later.
`customPermissions`	ProfileCustomPermissions[]	Indicates which custom permissions are available to users assigned to this profile. Available in API version 31.0 and later.
`description`	string	The profile description. Limit: 255 characters. Available in API version 30.0 and later.
`externalDataSourceAccesses`	ProfileExternalDataSourceAccess[]	Indicates which data sources with identity type of `Per User` are available to users assigned to this profile. Available in API version 27.0 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.
`layoutAssignments`	ProfileLayoutAssignments[]	Indicates which layout to use for this profile.

Field Name	Field Type	Description
`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 `allowRead` is `true`.
`pageAccesses`	ProfileApexPageAccess[]	Indicates which Visualforce pages that users assigned to this profile can execute.
`recordTypeVisibilities`	ProfileRecordTypeVisibility[]	Indicates the visibility of record types for users assigned to this profile. In API version 29.0 and later, this field is not retrieved or deployed for inactive record types.
`tabVisibilities`	ProfileTabVisibility[]	Indicates which record types are visible to a user assigned to this profile, and therefore which tabs within an app are visible.
`userLicense`	string	The `User License` for the profile. A user license determines the baseline of features that the user can access. Every user must have exactly one user license. 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.

ProfileApplicationVisibility

ProfileApplicationVisibility determines whether an app is visible to a user assigned to this profile.

Field Name	Field Type	Description
`application`	string	Required. The name of the app.
`default`	boolean	Required. Indicates whether the app is the default app (`true`) or not (`false`). Only one app per profile can be set to `true`.
`visible`	boolean	Required. Indicates whether this app is visible to users assigned to this profile (`true`) or not (`false`).

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 (`true`) or not (`false`).

ProfileCustomPermissions

ProfileCustomPermissions represents the custom permissions access for users assigned to a profile. Only enabled custom permissions are retrieved.

Field Name	Field Type	Description
`enabled`	boolean	Required. Indicates whether the custom permission is enabled (`true`) or not (`false`).
`name`	string	Required. The custom permission name.

ProfileExternalDataSourceAccess

ProfileExternalDataSourceAccess represents the data source access for users with identity type of `Per User`. Available in API version 27.0 and later.

Field Name	Field Type	Description
`enabled`	boolean	Required. Indicates whether the data source is enabled (`true`) or not (`false`).
`externalDataSource`	string	The name of the external data source.

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 (`true`) or not (`false`). In API version 30.0 and later, when deploying a new custom field, this field is `false` by default.
`field`	string	Required. Indicates the name of the field.

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

ProfileLayoutAssignments

ProfileLayoutAssignments determines which layout to use for a profile and a given entity.

Field Name	Field Type	Description
`layout`	string	Required. Indicates the layout for this particular entity.
`recordType`	string	This field is optional. If the `recordType` of the record matches a layout assignment rule, it will use the specified layout.

ProfileLoginHours

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

Field Name	Field Type	Description
`weekday``Start`	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. • Valid values for `weekday`: `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, or `sunday`. For example, `mondayStart` indicates the beginning of the login period for Monday. • Valid values for Start: the number of minutes since midnight. Must be evenly divisible by 60 (full hours). For example, `300` is 5:00 a.m.
`weekday``End`	string	Specifies the time on that day by which a user with this profile must log out. • Valid values for `weekday`: `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, or `sunday`. For example, `mondayEnd` indicates the close of the login period for Monday.

Field Name	Field Type	Description
		• Valid values for End: the number of minutes since midnight. Must be evenly divisible by 60 (full hours). For example, `1020` 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
`description`	string	Use this field to identify the purpose of the range, such as which part of a network corresponds to this range. This field is available in API version 31.0 and later.
`endAddress`	string	Required. The end IP address for the range.
`startAddress`	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
`allowCreate`	boolean	Indicates whether the object referenced by the `object` field can be created by the users assigned to this profile (`true`) or not (`false`). This field is named `revokeCreate` before version 14.0 and the logic is reversed. The field name change and the update from `true` to `false` 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 `true` to `false` and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.
`allowDelete`	boolean	Indicates whether the object referenced by the `object` field can be deleted by the users assigned to this profile (`true`) or not (`false`). This field is named `revokeDelete` before version 14.0 and the logic is reversed. The field name change and the update from `true` to `false` and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.

Field Name	Field Type	Description
`allowEdit`	boolean	Indicates whether the object referenced by the `object` field can be edited by the users assigned to this profile (`true`) or not (`false`). This field is named `revokeEdit` before version 14.0 and the logic is reversed. The field name change and the update from `true` to `false` and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.
`allowRead`	boolean	Indicates whether the object referenced by the `object` field can be seen by the users assigned to this profile (`true`) or not (`false`). This field is named `revokeRead` before version 14.0 and the logic is reversed. The field name change and the update from `true` to `false` and vice versa is automatically handled between versions and does not require any manual editing of existing XML component files.
`modifyAllRecords`	boolean	Indicates whether the object referenced by the `object` field can be read, edited, or deleted by the users assigned to this profile (`true`) or not (`false`), 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. 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 `modifyAllRecords` entries in Metadata API and don't return an error if "Modify All Data" is enabled on the profile.
`object`	string	Required. The name of the object whose permissions are altered by this profile, for example, `MyCustomObject__c`.
`viewAllRecords`	boolean	Indicates whether the object referenced by the `object` field can be read by the users assigned to this profile (`true`) or not (`false`), 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. 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 `viewAllRecords` entries in the Metadata API and don't return an error if "View All Data" is enabled on the profile.

ProfileApexPageAccess

ProfileApexPageAccess determines which Visualforce pages that users assigned to this profile can execute.

Field Name	Field Type	Description
`apexPage`	string	Required. The Visualforce page name.
`enabled`	boolean	Required. Indicates whether users assigned to this profile can execute the Visualforce page (`true`) or not (`false`).

ProfileRecordTypeVisibility

ProfileRecordTypeVisibility represents the visibility of record types for this profile. Record types let you offer different business processes, picklist values, and page layouts to different users.

Field Name	Field Type	Description
`default`	boolean	Required. Indicates whether the record type is the default for this pair of profile and object (`true`) or not (`false`). Only one default is allowed per object.
`personAccountDefault`	boolean	Indicates whether the record type is the default person account record type for this pair of profile and object (`true`) or not (`false`). Only one person account record type default is allowed per object. This field is only relevant for record types for account or contact objects. For more information about person accounts, see "What Is a Person Account?" in the Salesforce online help. Person accounts are not enabled by default in Salesforce. To request person accounts, contact Salesforce.
`recordType`	string	Required. The record type name, for example `Account.MyRecordType`.
`visible`	boolean	Required. Indicates whether this record type is visible to users assigned to this profile (`true`) or not (`false`).

ProfileTabVisibility

ProfileTabVisibility represents the visibility of tabs for this profile. For version 17.0 and later, ProfileTabVisibility supports visibility of tabs for standard objects. The manifest file must include the standard object corresponding to a standard tab to retrieve the tab visibility in a profile.

Field Name	Field Type	Description
`tab`	string	Required. The name of the tab.
`visibility`	TabVisibility (enumeration of type string)	Required. Indicates the visibility of the tab. Valid values are: • `DefaultOff`—The tab is available on the All Tabs page. Individual users can customize their display to make the tab visible in any app. • `DefaultOn`—The tab is available on the All Tabs page and appears in the visible tabs for its associated app. Individual users can customize their display to hide the tab or make it visible in other apps.

Field Name	Field Type	Description
		• `Hidden`—The tab isn't available on the All Tabs page or visible in any apps.

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 (`true`) or disabled (`false`).
`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();
  }
}
```

Declarative Metadata Sample Definition

The following is the definition of a profile in an organization with a custom app, custom object, record type, tab, and user permission:

```
<?xml version="1.0" encoding="UTF-8"?>
<Profile xmlns="http://soap.sforce.com/2006/04/metadata">
    <applicationVisibilities>
        <application>PubApps__Myriad_Publishing</application>
        <default>false</default>
        <visible>true</visible>
    </applicationVisibilities>
    <custom>true</custom>
    <objectPermissions>
        <object>TestWeblinks__c</object>
    </objectPermissions>
    <recordTypeVisibilities>
        <default>true</default>
        <recordType>TestWeblinks__c.My First Recordtype</recordType>
        <visible>true</visible>
    </recordTypeVisibilities>
    <tabVisibilities>
        <tab>Myriad Publications</tab>
        <visibility>DefaultOn</visibility>
    </tabVisibilities>
    <userPermissions>
        <enabled>true</enabled>
        <name>APIEnabled</name>
    </userpermissions>
</Profile>
```

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>
   <types>
        <members>*</members>
        <name>Profile</name>
    </types>
    <version>36.0</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>36.0</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>36.0</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 (`true`) or not (`false`) 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 `fullName` 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.
`queueSobject`	QueueSobject[]	Indicates the supported entity types.

QueueSobject

QueueSobject represents an entity type that the queue supports.

Field Name	Field Type	Description
sobjectType	string	Valid values are: • Case • Lead • ServiceContract • Custom objects (e.g. ObjA_c)

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>
    <queueSobject>
        <sobjectType>Case</sobjectType>
    </queueSobject>
    <queueSobject>
        <sobjectType>Lead</sobjectType>
    </queueSobject>
    <queueSobject>
        <sobjectType>ObjA_c</sobjectType>
    </queueSobject>
</Queue>
```

QuickAction

Represents a specified create or update quick 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

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
`canvas`	string	If a custom action is created using a canvas app, the app name. Returns the fully qualified name of the canvas app in the format `<namespace>__<dev_name>`, if the quick action type is `Canvas`; otherwise, returns null. This field is available in API version 29.0 and later.
`description`	string	The description of the action.
`fieldOverrides`	FieldOverride	The specific field that may be overridden within a QuickAction.
`height`	int	If a custom action is created, this field represents the height in pixels of the action pane.
`icon`	string	The icon used to identify the action. API version 32.0 and later returns different icons than in earlier API versions.
`isProtected`	boolean	Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`label`	string	Identifies the action and displays to users. This is also the default identifier used for the API and managed packages.
`optionsCreateFeedItem`	boolean	Indicates whether successful completion of the action creates a feed item (`true`) or not (`false`). Applies only to Create Record, Update Record, and Log a Call quick action types. Available in API version 36.0 and later.
`page`	string	If a custom action is created using a Visualforce page, this field identifies the page.
`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: • `LogACall` • `LogANote` • `New` (A new record)

Field Name	Field Type	Description
		• `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") • `SendEmail` (This value is available in API version 31.0 and later.) • `QuickRecordType` • `Quick` (A quick record)
`successMessage`	string	The message that displays to the user upon successful completion of the action. Available in API version 36.0 and later.
`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: • Business Account • Person Account • Master
`type`	QuickActionType (enumeration of type string)	The type of quick action. Valid values are: • `Create` • `VisualforcePage` • `Post` • `SendEmail` (This value is available in API version 31.0 and later.) • `LogACall` • `SocialPost` • `Canvas` • `Update`

Field Name	Field Type	Description
`width`	int	If a custom action is created, this field represents 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 affect user efficiency.

Field Name	Field Type	Description
`layoutSectionStyle`	LayoutSectionStyle (enumeration of type string)	The type of layout structure used. The valid values are: • `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 (`true`) or not (`false`).
`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 affect user efficiency.
`uiBehavior`	UiBehavior (enumeration of type string)	Specifies user input behavior for specific fields in QuickActionLayoutItem. The valid values are: • `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>testActionDefinitionTypesCreateTask</description>
    <label>testActionDefinitionTypesCreateTask</label>
    <optionsCreateFeedItem>true</optionsCreateFeedItem>
    <quickActionLayout>
        <layoutSectionStyle>TwoColumnsLeftToRight</layoutSectionStyle>
        <quickActionLayoutColumns>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>OwnerId</field>
                <uiBehavior>Required</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>WhoId</field>
                <uiBehavior>Edit</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>WhatId</field>
                <uiBehavior>Edit</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>ActivityDate</field>
                <uiBehavior>Edit</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>Subject</field>
                <uiBehavior>Edit</uiBehavior>
```

```
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>Status</field>
                <uiBehavior>Required</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>Priority</field>
                <uiBehavior>Required</uiBehavior>
            </quickActionLayoutItems>
        </quickActionLayoutColumns>
        <quickActionLayoutColumns/>
    </quickActionLayout>
    <successMessage>This is a success message</successMessage>
    <targetObject>Task</targetObject>
    <targetParentField>What</targetParentField>
    <type>Create</type>
</QuickAction>
```

RemoteSiteSetting

Represents a remote site setting. Before any Visualforce page, Apex callout, or JavaScript code using XmlHttpRequest in an s-control or custom button can call an external site, that site must be registered in the Remote Site Settings page, or the call will fail. 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	Required. Indicates whether code within Salesforce can access the remote site regardless of whether the user's connection is over HTTP or HTTPS (`true`) or not (`false`). When `true`, code within Salesforce can pass data from an HTTPS session to an HTTP session, and vice versa.

Field	Field Type	Description
		Warning: Only set to `true` if you understand the security implications.
`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.
`isActive`	boolean	Required. Indicates if the remote site setting is active (`true`) or not (`false`).
`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>
```

Report

Represents a custom report. This type extends the Metadata metadata type and inherits its `fullName` field. This metadata type only supports custom reports; standard reports are not supported.

Declarative Metadata File Suffix and Directory Location

Reports are stored in the `reports` directory of the corresponding package directory. The file name matches the report title and the extension is `.report`.

Retrieving Reports

You can't use the wildcard (*) symbol with reports in `package.xml`. To retrieve the list of reports for populating `package.xml` with explicit names, call `listMetadata()` and pass in `ReportFolder` as the type. Note that ReportFolder is not returned as a type in `describeMetadata()`. Report is returned from `describeMetadata()` with an associated attribute of `inFolder` set to true. If that attribute is set to true, you can construct the type by using the component name with the word Folder, such as ReportFolder.

The following example shows folders in `package.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
        <name>Report</name>
    </types>
    <version>36.0</version>
</Package>
```

Version

Report components are available in API version 14.0 and later.

Fields

The following information assumes that you are familiar with creating and running reports. For more information on these fields, see "Create a Report" in the Salesforce online help.

Field	Field Type	Description
`aggregates`	ReportAggregate[]	List that defines custom summary formulas for summary, matrix, and joined reports.
`block`	Report[]	Represents each block in a joined report where every block can be of a different report type.
`blockInfo`	ReportBlockInfo	Defines attributes for each block in a joined report.
`buckets`	ReportBucketField[]	Defines a bucket field to be used in the report. This field is available in API version 24.0 and later.
`chart`	ReportChart	Defines a chart for summary and matrix reports .
`colorRanges`	ReportColorRange[]	List that specifies conditional highlighting for report summary data.

Field	Field Type	Description
`columns`	ReportColumn[]	List that specifies the fields displayed in the report. Fields appear in the report in the same order as they appear in the Metadata API file.
`crossFilters`	ReportCrossFilter[]	Defines a cross filter's object, related object, and condition (WITH or WITHOUT). This field is available in API version 36.0 and later.
`currency`	`CurrencyIsoCode` (enumeration of type string)	When using multiple currencies, some reports allow you to display converted amounts by selecting the appropriate column to display. For example, in opportunity reports, you can include the Amount (converted) column on the report. This field is an enumeration of type string that defines the currency in which to display converted amounts. Valid values: Must be one of the valid alphabetic, three-letter currency ISO codes defined by the ISO 4217 standard, such as `USD`, `GBP`, or `JPY`.
`description`	string	Specifies a general description, which is displayed with the report name. Maximum characters: 255 characters.
`division`	string	If your organization uses divisions to segment data and you have the "Affected by Divisions" permission, records in the report must match this division. This field is available in API version 17.0 and later.

`filter`	ReportFilter	Limits report results to records with specific data. For example, you can limit report results to opportunities for which the amount is greater than $1,000:
		`<filter>`
		`<criteriaItems>`
		`<column>AMOUNT</column>`
		`<operator>greaterThan</operator>`
		`<value>1000</value>`
		`</criteriaItems>`
		`</filter>`
		For more information, see "Enter Filter Criteria" in the Salesforce online help.

`folderName`	string	Name of the folder that houses the report. This field is available in API version 35.0 and later.

Field	Field Type	Description
`format`	ReportFormat (enumeration of type string)	Defines the report format. For example, `Tabular` for a simple data list without subtotals.
`fullName`	string	The report unique developer name used as an 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`groupingsAcross`	ReportGrouping[]	List that defines the fields by which you want to group and subtotal data across a matrix report (row headings). When grouping by a date field, you can further group the data by a specific time period such as days, weeks, or months. Maximum: 2 fields.
`groupingsDown`	ReportGrouping[]	For Summary and Matrix reports: List that defines the fields by which you want to group and subtotal. For summary reports, choosing more than one sort field allows you to subsort your data. For matrix reports, specifies summary fields for column headings. When grouping by a date field, you can further group the data by a specific time period such as days, weeks, or months. Maximum for matrix reports: 2. Maximum for summary reports: 3
`historicalSelector`	ReportHistoricalSelector	Defines a date range for which historical trend reporting data is to be captured. Default is "Any Historical Date." Available in API version 29 and later.
`name`	string	Required. The report name. For example, `Opportunity Pipeline`
`params`	ReportParam[]	List that specifies settings specific to each report type, in particular options that let you filter a report to obtain useful subsets. For example, the Activities report type lets you specify whether you want to see open or closed activities or both and whether you want to see tasks or events or both. Valid values depend on the report type.
`reportType`	string	Required. Defines the type of data in the report. For example, `Opportunity` to create a report of opportunities data.

Field	Field Type	Description
`roleHierarchyFilter`	string	The role name for a report drill down. Some reports, such as opportunity and activity reports, display Hierarchy links that allow you to drill down to different data sets based on the role hierarchy. This field is available in API version 17.0 and later.
`rowLimit`	int	Defines the maximum number of rows that can be returned for the report.
`scope`	string	Defines the scope of data on which you run the report. For example, whether you want to run the report against all opportunities, opportunities you own, or opportunities your team owns. Valid values depend on the `reportType`. For example, for Accounts reports: • `MyAccounts` • `MyTeamsAccounts` • `AllAccounts`
`showCurrentDate`	boolean	Can be set to `true` for historical trending reports in matrix format. Available in API version 29 and later.
`showDetails`	boolean	`false` shows a collapsed view of the report with only the headings, subtotals, and total. Default: `true`
`sortColumn`	string	Specifies the field on which to sort data in the report. Use `sortOrder` to specify the sort order.
`sortOrder`	SortOrder (enumeration of type string)	Specifies the sort order. Use `sortColumn` to specify the field on which to sort.
`territoryHierarchyFilter`	string	The territory name for a report drill down. If your organization uses territory management, some reports display Hierarchy links that allow you to drill down to different data sets based on the territory hierarchy. This field is available in API version 17.0 and later.
`timeFrameFilter`	ReportTimeFrameFilter	Limits report results to records within a specified time frame.
`userFilter`	string	The user name for a report drill down. Some reports, such as opportunity and activity reports,

Field	Field Type	Description
		display Hierarchy links that allow you to drill down to different data sets based on the user hierarchy. This field is available in API version 17.0 and later.

ReportAggregate

ReportAggregate defines custom summary formulas on summary, matrix, and joined reports. For more information on these fields, see "Build a Custom Summary Formula" in the Salesforce online help.

Field	Field Type	Description
`acrossGroupingContext`	string	Defines the row grouping level at which you want your custom summary formula to be displayed. This is a new field in API version 15.0.
`calculatedFormula`	string	Required. The custom summary formula. For example, `AMOUNT:SUM + OPP_QUANTITY:SUM`
`datatype`	ReportAggregateDatatype (enumeration of type string)	Required. Specifies the data type for formatting and display of the custom summary formula results.
`description`	string	The custom summary formula description. Maximum: 255 characters.
`developerName`	string	Required. The internal development name of the custom summary formula, for example, `FORMULA1`. This is used to reference custom summary formulas from other report components, including conditional highlighting.
`downGroupingContext`	string	Defines the column grouping level at which you want your custom summary formula to be displayed. This field is available in API version 15.0 and later.
`isActive`	booleanblo	Required. `true` displays the formula result in the report. `false` does not display the result in the report.
`isCrossBlock`	boolean	Determines whether the custom summary formula is a cross-block formula, which is available with joined reports. `true` indicates a cross-block custom summary formula. `false` indicates a standard custom summary formula. This field is available in API version 25.0 and later.
`masterLabel`	string	Required. The custom summary formula label (name).
`reportType`	string	Required for joined reports. Specifies the `reportType` of the blocks to which the `aggregate` can be added.
`scale`	int	The formula result is calculated to the specified number of decimal places. Valid values `0` through `18`.

ReportBlockInfo

ReportBlockInfo defines blocks in a joined report.

Field	Field Type	Description
`aggregateReferences`	ReportAggregateReference[]	Lists the `aggregates` that represent the custom summary formulas used in a joined report block.
`blockId`	string	Required. `blockId` is used in cross-block custom summary formulas and joined report charts to identify the block containing each summary field. `blockId` is assigned automatically. Valid values are B1 through B5. This field is available in API version 25.0 and later.
`joinTable`	string	Required. Refers to the entity used to join blocks in a joined report. The entity provides a list of fields that are available for globally grouping across the blocks.

ReportAggregateReference

ReportAggregateReference defines the developer name used for custom summary formulas in joined reports.

Field	Field Type	Description
`aggregate`	string	Required. The `developerName` of the ReportAggregate, which specifies the custom summary formula used in a block of a joined report.

ReportBucketField

ReportBucketField defines a bucket to be used in the report.

Field	Field Type	Description
`bucketType`	ReportBucketFieldType (enumeration of type string)	Required. Specifies the type of bucket. Valid values: • text • number • picklist
`developerName`	string	Required. A unique name used as the `<field>` value to display a bucket field in the column list and other report components, including sort, filter, list, group, and chart. Must be of the format `BucketField_`***`name`***. For example, `BucketField_BusinessSize`.
`masterLabel`	string	Required. The bucket field label. Maximum 40 characters. Any line breaks, tabs, or multiple spaces at the beginning or end of

Field	Field Type	Description
		the label are removed. Any of these characters within the label are reduced to a single space.
`nullTreatment`	ReportBucketFieldNullTreatment (enumeration of type string)	For numeric bucket fields only. Specifies whether empty values are treated as zeros (`z`) or not (`n`).
`otherBucketLabel`	string	The label of the container for unbucketed values.
`sourceColumnName`	string	Required. The source field that the bucket is applied to. For example, `SALES` or `INDUSTRY`.
`values`	ReportBucketFieldValue (enumeration of type string)	Defines one bucket value used in the bucket field. Note: While this name is plural, it represents a single bucket. In typical use, a bucket field contains multiple buckets.

ReportBucketFieldValue

ReportBucketFieldValue defines a bucket value used in the bucket field.

Field	Field Type	Description
`sourceValues`	ReportBucketFieldSourceValue (enumeration of type string)	The value of a bucket in the bucket field. Valid values: • `sourceValue`—Used for picklist and text bucket fields. For picklists, describes the picklist item in the bucket. For example, the sourceValue of a bucket on `TYPE` could be `Customer`. For text, the full string for the item in the bucket. For example, the sourceValue of a bucket on `ADDRESS_STATE1` could be `NY`. • `from`—Used only on numeric bucket fields. A non-inclusive lower bound for a numeric bucket range. This value must be a number. • `to`—Used only on numeric bucket fields. The inclusive upper bound for a numeric bucket range. This value must be a number. In numeric buckets, the first value must only have `to` and last value must only have `from`. All other values must have both `to` and `from`.
`value`	string	Required. The name of a specific bucket value within the bucket field.

ReportGrouping

ReportGrouping defines how to group, subtotal, and sort data for summary, matrix, and joined reports.

Field	Field Type	Description
`aggregateType`	ReportAggrType (enumeration of type string)	The type of aggregate value to sort by. Valid values are: • `Sum` • `Average` • `Maximum` • `Minimum` • `RowCount`
`dateGranularity`	UserDateGranularity (enumeration of type string)	When grouping by a date field, the time period by which to group.
`field`	string	Required. The field by which you want to summarize data. For example, `CAMPAIGN_SOURCE`
`sortByName`	string	The API name of the column, aggregate or custom summary field used to order the grouping.
`sortOrder`	SortOrder	Required. Whether to sort data in ascending or descending alphabetical and numerical order.
`sortType`	ReportSortType (enumeration of type string)	Indicates if the grouping is sorted by a column, aggregate or custom summary field. Valid values are: • `Column` • `Aggregate` • `CustomSummaryFormula`

ReportHistoricalSelector

ReportHistoricalSelector defines a date range for historical data.

Field	Field Type	Description
`snapshot`	string	Represents the date value to apply a historical filter, either relative (in the format `N_DAYS_AGO:2`) or absolute (in the format `yyyy-MM-dd`). If unspecified, it's assumed that the filter will be applied to all the columns the user sees. Available in API version 29 and later.

SortOrder

An enumeration of type string that defines the order in which data is sorted in the report fields. Valid values:

Field	Description
`Asc`	Sorts data in ascending alphabetical and numerical order.

Field	Description
`Desc`	Sorts data in descending alphabetical and numerical order.

UserDateGranularity

An enumeration of type string that defines the time period by which to group data. Valid values:

Enumeration Value	Description
`None`	No grouping by date
`Day`	By day
`Week`	By week
`Month`	By month
`Quarter`	By quarter
`Year`	By year
`FiscalQuarter`	By fiscal quarter. You can set the fiscal year for your organization. See “Set the Fiscal Year” in the Salesforce online help.
`FiscalYear`	By fiscal year
`MonthInYear`	By calendar month in year
`DayInMonth`	By calendar day in month
`FiscalPeriod`	When custom fiscal years are enabled: By fiscal period
`FiscalWeek`	When custom fiscal years are enabled: By fiscal week

ReportSummaryType

An enumeration of type string that defines how report fields are summarized. Valid values:

Enumeration Value	Description
`Sum`	Total
`Average`	Average
`Maximum`	Largest value
`Minimum`	Smallest value
`None`	The field is not summarized.

ReportColorRange

ReportColorRange defines conditional highlighting for report summary data.

Field	Field Type	Description
`aggregate`	ReportSummaryType (enumeration of type string)	Required. Defines how the field specified in `columnName` is summarized. For example, `Sum`.
`columnName`	string	Required. Specifies the field whose value ranges are represented by colors.
`highBreakpoint`	double	Required. Specifies the number that separates the mid color from the high color.
`highColor`	string	Required. Specifies the color (in HTML format) to represent data that falls into the high number range. This color spans from the `highBreakpoint` value.
`lowBreakpoint`	double	Required. Specifies the number that separates the low color from the mid color.
`lowColor`	string	Required. Specifies a color (in HTML format) to represent data that falls into the low value range, below the `lowBreakpoint` value.
`midColor`	string	Required. Specifies a color (in HTML format) to represent data that falls into the mid value range.

ReportColumn

ReportColumn defines how fields (columns) are displayed in the report.

Field	Field Type	Description
`aggregateTypes`	ReportSummaryType[] (enumeration of type string)	List that defines if and how each report field is summarized.
`field`	string	Required. The field name. For example, `AGE` or `OPPORTUNITY_NAME`
`reverseColors`	boolean	In historical trend reports, displays greater Date values as green and greater Amount values as red, reversing the default colors. Available in API version 29.0 and later.
`showChanges`	boolean	In historical trend reports, adds a column displaying the difference between current and historical Date and Amount values. Available in API version 29.0 and later.

ReportFilter

ReportFilter limits the report results by filtering data on specified fields.

Field	Field Type	Description
booleanFilter	string	Specifies filter logic conditions. For more information on filter logic, see "Getting the Most Out of Filter Logic" in the Salesforce online help.

criteriaItems	ReportFilterItem	The criteria by which you want to filter report data, either by comparing historical values or by applying a date range.
		`<criteriaItems>`
		`criteriaItems ReportFilterItem`
		`<column>Opportunity.Opportunity__hd$Amount__hst</column>`
		`<columnToColumn>false</columnToColumn>`
		`<operator>equals</operator>`
		`<snapshot>N_DAYS_AGO:90</snapshot>`
		`<value>100</value>`
		`</criteriaItems>`

language	Language (enumeration of type string)	The language used when a report filters against a picklist value using the operators `contains` or `startsWith`. For a list of valid language values, see Language.

ReportFilterItem

ReportFilterItem limits the report results by filtering data on specified fields.

Field	Field Type	Description
column	string	Required. The field on which you want to filter data. For example, `AMOUNT`
columnToColumn	boolean	Indicates that the field contains data from a historical snapshot. Available in API version 29 and later.
operator	FilterOperation (enumeration of type string)	Required. An enumeration of type string that defines the operator used to filter the data, for example, `greaterThan`. For valid values, see FilterOperation.
snapshot	string	Represents the date value, either relative (in the format `N_DAYS_AGO:2`) or absolute (in the format `yyyy-MM-dd`). Available in API version 29 and later.
value	string	The value by which you want to filter the data, for example, `1000`. Note that the Metadata API filter condition values do not always match those that you enter in the report wizard. For example, in the Metadata API dates are always converted to the US date format and values entered in a non-US English language may be converted to a standard US English equivalent.

ReportFormat

An enumeration of type string that defines the report format. Valid values:

Enumeration Value	Description
`Matrix`	Summarizes data in a grid. Use to compare related totals.
`Summary`	Lists, sorts, and subtotals data.
`Tabular`	Lists data with no sorting or subtotals.
`Joined`	Joins data from different report types storing each report's data in its own block.

ReportParam

ReportParam represents settings specific to a report type, especially options that let you filter a report to certain useful subsets.

Field	Field Type	Description
`name`	string	Required. Specifies a specific `reportType` setting.
`value`	string	Required. The setting value.

ReportAggregateDatatype

An enumeration of type string that specifies the data type for formatting and display of custom summary formula results. Valid values:

Enumeration Value
`currency`
`number`
`percent`

ReportChart

ReportChart represents charts on summary, matrix, and joined reports.

Field	Field Type	Description
`backgroundColor1`	string	Specifies the beginning color (in HTML format) for a gradient color background.
`backgroundColor2`	string	Specifies the end color (in HTML format) for a gradient color background.
`backgroundFadeDir`	ChartBackgroundDirection (enumeration of type string)	Specifies the direction for a gradient color background. Use with `backgroundColor1` to specify the beginning color and `backgroundColor2` to specify the end color for the gradient

Field	Field Type	Description
		design. Use white for both if you do not want a background design. Valid values: • `diagonal` • `leftToRight` • `topToBottom`
`chartSummaries`	ChartSummary[]	Specifies the summaries you want to use for the chart. Invalid summaries are ignored without notification. If there are no valid summaries, RowCount is used by default for the axis value. This field is available in API version 17.0 and later.
`chartType`	ChartType (enumeration of type string)	Required. Specifies the chart type. Available chart types depend on the report type.
`enableHoverLabels`	boolean	Specifies whether to display values, labels, and percentages when hovering over charts. Hover details depend on chart type. Percentages apply to pie, donut, and funnel charts only. This field is available in API version 17.0 and later.
`expandOthers`	boolean	Specifies whether to combine all groups less than or equal to 3% of the total into a single 'Others' wedge or segment. This only applies to pie, donut, and funnel charts. Set to `true` to show all values individually on the chart; set to `false` to combine small groups into 'Others.' This field is available in API version 17.0 and later.
`groupingColumn`	string	Specifies the field by which to group data. This data is displayed on the X-axis for vertical column charts and on the Y-axis for horizontal bar charts.
`legendPosition`	ChartLegendPosition (enumeration of type string)	Required. The location of the legend with respect to the chart. The valid values are: • `Bottom` • `OnChart` • `Right`
`location`	ChartPosition (enumeration of type string)	Required. Specifies whether the chart is displayed at the top or bottom of the report.
`secondaryGroupingColumn`	string	For grouped chart types: Specifies the field by which to group the data.
`showAxisLabels`	boolean	For bar and line charts: Specifies whether the chart displays names for each axis.
`showPercentage`	boolean	Indicates if percentages are displayed for wedges and segments of pie, donut, and funnel charts, as well as for gauges (`true`), or not (`false`).

Field	Field Type	Description
showTotal	boolean	Indicates if the total is displayed for donut charts and gauges (true), or not (false).
showValues	boolean	Indicates if the values of individual records or groups are displayed for charts (true), or not (false).
size	ReportChartSize (enumeration of type string)	Required. Specifies the chart size.
summaryAggregate	ReportSummaryType (enumeration of type string)	Defines how to summarize the chart data. For example, Sum. No longer supported in version API 17.0 and later. See chartSummaries.
summaryAxisManualRangeEnd	double	When specifying the axis range manually: Defines the ending value.
summaryAxisManualRangeStart	double	When specifying the axis range manually: Defines the starting value.
summaryAxisRange	ChartRangeType (enumeration of type string)	Required. For bar, line, and column charts: Defines whether to specify the axis range manually or automatically.
summaryColumn	string	Required. Specifies the field by which to summarize the chart data. Typically this field is displayed on the Y-axis. No longer supported in version API 17.0 and later. See chartSummaries.
textColor	string	The color (in HTML format) of the chart text and labels.
textSize	int	The size of the chart text and labels. Valid values: • 8 • 9 • 10 • 12 • 14 • 18 • 24 • 36 The maximum size is 18. Larger values are shown at 18 points.
title	string	The chart title. Max 255 characters.
titleColor	string	The color (in HTML format) of the title text.
titleSize	int	The size of the title text. Valid values: • 8 • 9 • 10 • 12

Field	Field Type	Description
		• 14 • 18 • 24 • 36 The maximum size is 18. Larger values are shown at 18 points.

ChartType

An enumeration of type string that defines the chart type. For information on each of these chart types, see "Chart Types" in the Salesforce online help. Valid values:

Enumeration Value
None
HorizontalBar
HorizontalBarGrouped
HorizontalBarStacked
HorizontalBarStackedTo100
VerticalColumn
VerticalColumnGrouped
VerticalColumnStacked
VerticalColumnStackedTo100
Line
LineGrouped
LineCumulative
LineCumulativeGrouped
Pie
Donut
Funnel
Scatter
ScatterGrouped
VerticalColumnLine
VerticalColumnGroupedLine
VerticalColumnStackedLine

Enumeration Value	
`Plugin`	Reserved for future use. This value is available in API version 31.0 and later.

ChartPosition

An enumeration of type string that specifies the position of the chart in the report. Valid values:

Enumeration Value
`CHART_TOP`
`CHART_BOTTOM`

ChartSummary

ChartSummary defines how data in the chart is summarized. Valid values:

Field	Field Type	Description
`aggregate`	ReportSummaryType	Specifies the aggregation method—such as `Sum`, `Average`, `Min`, and `Max`—for the summary value. Use the `column` field to specify the summary value to use for the aggregation. You don't need to specify this field for RowCount or custom summary formulas.
`axisBinding`	ChartAxis	Specifies the axis or axes to use on the chart. Use the `column` field to specify the summary value to use for the axis.
`column`	string	Required. Specifies the summary field for the chart data. If all columns are invalid, RowCount is used by default for the axis value. For vertical column and horizontal bar combination charts, you can specify up to four values.

ChartAxis

An enumeration of type string that specifies the axis or axes to be used in charts. Valid values:

Enumeration Value	Description
`x`	The summary value to use for the X-axis of a scatter chart.
`y`	The Y-axis for the chart.
`y2`	The secondary Y-axis for vertical column combination charts with a line added.

ReportChartSize

An enumeration of type string that specifies the chart size. Valid values:

Enumeration Value
`Tiny`
`Small`
`Medium`
`Large`
`Huge`

ChartRangeType

An enumeration of type string that defines the report format. Valid values:

Enumeration Value
`Auto`
`Manual`

ReportTimeFrameFilter

ReportTimeFrameFilter represents the report time period.

Field	Field Type	Description
`dateColumn`	string	Required. The date field on which to filter data. For example, `CLOSE_DATE`
`endDate`	date	When `interval` is `INTERVAL_CUSTOM`, specifies the end of the custom time period.
`interval`	UserDateInterval (enumeration of type string)	Required. Specifies the period of time.
`startDate`	date	When `interval` is `INTERVAL_CUSTOM`, specifies the start of the custom time period.

ReportCrossFilter

ReportCrossFilter represents the cross filter functionality in reports.

Field	Field Type	Description
`criteriaItems`	ReportFilterItem	Represents the subfilters of a cross filter. There can be up to five subfilters. This field requires the following attributes. • `Column` • `Operator` • `Value`
`operation`	ObjectFilterOperator. Enumeration of type string	The action indicating whether to include or exclude an object. Valid values: `with` and `without`.
`primaryTableColumn`	string	The parent object used for the cross filter.
`relatedTable`	string	The child object used for the cross filter.
`relatedTableJoinColumn`	string	The field from the child object that is used to join the parent.

Declarative Metadata Sample Definition

A sample XML snippet using cross filters to build an Accounts report for cases where case status is not closed:

```
    <crossFilters>
        <criteriaItems>
            <column>Status</column>
            <operator>notequal</operator>
            <value>Closed</value>
        </criteriaItems>
        <operation>with</operation>
        <primaryTableColumn>ACCOUNT_ID</primaryTableColumn>
        <relatedTable>Case</relatedTable>
        <relatedTableJoinColumn>Account</relatedTableJoinColumn>
    </crossFilters>
```

Note: This sample was generated using the API version 23.0.

UserDateInterval

An enumeration of type string that defines the period of time. Valid values:

Enumeration Value	Description
`INTERVAL_CURRENT`	Current fiscal quarter
`INTERVAL_CURNEXT1`	Current and next fiscal quarters
`INTERVAL_CURPREV1`	Current and previous fiscal quarters
`INTERVAL_NEXT1`	Next fiscal quarter
`INTERVAL_PREV1`	Previous fiscal quarter
`INTERVAL_CURNEXT3`	Current and next three fiscal quarters

Enumeration Value	Description
INTERVAL_CURFY	Current fiscal year
INTERVAL_PREVFY	Previous fiscal year
INTERVAL_PREV2FY	Previous two fiscal years
INTERVAL_AGO2FY	Two fiscal years ago
INTERVAL_NEXTFY	Next fiscal year
INTERVAL_PREVCURFY	Current and previous fiscal years
INTERVAL_PREVCUR2FY	Current and previous two fiscal years
INTERVAL_CURNEXTFY	Current and next fiscal year
INTERVAL_CUSTOM	A custom time period. Use `startDate` and `endDate` fields to specify the time period's start date and end date.
INTERVAL_YESTERDAY	Yesterday
INTERVAL_TODAY	Today
INTERVAL_TOMORROW	Tomorrow
INTERVAL_LASTWEEK	Last calendar week
INTERVAL_THISWEEK	This calendar week
INTERVAL_NEXTWEEK	Next calendar week
INTERVAL_LASTMONTH	Last calendar month
INTERVAL_THISMONTH	This calendar month
INTERVAL_NEXTMONTH	Next calendar month
INTERVAL_LASTTHISMONTH	Current and previous calendar months
INTERVAL_THISNEXTMONTH	Current and next calendar months
INTERVAL_CURRENTQ	Current calendar quarter
INTERVAL_CURNEXTQ	Current and next calendar quarters
INTERVAL_CURPREVQ	Current and previous calendar quarters
INTERVAL_NEXTQ	Next calendar quarter
INTERVAL_PREVQ	Previous calendar quarter
INTERVAL_CURNEXT3Q	Current and next three calendar quarters
INTERVAL_CURY	Current calendar year
INTERVAL_PREVY	Previous calendar year
INTERVAL_PREV2Y	Previous two calendar years
INTERVAL_AGO2Y	Two calendar years ago

Enumeration Value	Description
INTERVAL_NEXTY	Next calendar year
INTERVAL_PREVCURY	Current and previous calendar years
INTERVAL_PREVCUR2Y	Current and previous two calendar years
INTERVAL_CURNEXTY	Current and next calendar years
INTERVAL_LAST7	Last 7 days
INTERVAL_LAST30	Last 30 days
INTERVAL_LAST60	Last 60 days
INTERVAL_LAST90	Last 90 days
INTERVAL_LAST120	Last 120 days
INTERVAL_NEXT7	Next 7 days
INTERVAL_NEXT30	Next 30 days
INTERVAL_NEXT60	Next 60 days
INTERVAL_NEXT90	Next 90 days
INTERVAL_NEXT120	Next 120 days
LAST_FISCALWEEK	When custom fiscal years are enabled: Last fiscal week
THIS_FISCALWEEK	When custom fiscal years are enabled: This fiscal week
NEXT_FISCALWEEK	When custom fiscal years are enabled: Next fiscal week
LAST_FISCALPERIOD	When custom fiscal years are enabled: Last fiscal period
THIS_FISCALPERIOD	When custom fiscal years are enabled: This fiscal period
NEXT_FISCALPERIOD	When custom fiscal years are enabled: Next fiscal period
LASTTHIS_FISCALPERIOD	When custom fiscal years are enabled: This fiscal period and last fiscal period
THISNEXT_FISCALPERIOD	When custom fiscal years are enabled: This fiscal period and next fiscal period
CURRENT_ENTITLEMENT_PERIOD	Current entitlement period
PREVIOUS_ENTITLEMENT_PERIOD	Previous entitlement period
PREVIOUS_TWO_ENTITLEMENT_PERIODS	Previous two entitlement periods
TWO_ENTITLEMENT_PERIODS_AGO	Two entitlement periods ago
CURRENT_AND_PREVIOUS_ENTITLEMENT_PERIOD	Current and previous entitlement period
CURRENT_AND_PREVIOUS_TWO_ENTITLEMENT_PERIODS	Current and previous two entitlement periods

Declarative Metadata Sample Definition

A sample XML report definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Report xmlns="http://soap.sforce.com/2006/04/metadata">
    <aggregates>
        <acrossGroupingContext>CRT_Object__c$Id</acrossGroupingContext>
        <calculatedFormula>PREVGROUPVAL(CRT_Object__c.Currency__c:AVG, CRT_Object__c.Id)
*
                PARENTGROUPVAL(CRT_Object__c.Number__c:MAX, CRT_Object__c.CreatedBy.Name,

                COLUMN_GRAND_SUMMARY)/RowCount</calculatedFormula>
        <datatype>number</datatype>
        <developerName>FORMULA1</developerName>
        <downGroupingContext>CRT_Object__c$CreatedBy</downGroupingContext>
        <isActive>true</isActive>
        <masterLabel>CurrCSF</masterLabel>
        <scale>2</scale>
    </aggregates>
    <aggregates>
        <acrossGroupingContext>CRT_Object__c$LastModifiedDate</acrossGroupingContext>
        <calculatedFormula>IF(RowCount&gt;10,
                BLANKVALUE(ROUND(PREVGROUPVAL(CRT_Object__c.Currency__c:SUM,
                CRT_Object__c.LastModifiedDate),3),
                PARENTGROUPVAL(CRT_Object__c.Number__c:SUM, ROW_GRAND_SUMMARY,
                CRT_Object__c.Id))  , 1000)</calculatedFormula>
        <datatype>number</datatype>
        <developerName>FORMULA2</developerName>
        <downGroupingContext>GRAND_SUMMARY</downGroupingContext>
        <isActive>true</isActive>
        <masterLabel>numCSF</masterLabel>
        <scale>2</scale>
    </aggregates>
    <buckets>
        <bucketType>number</bucketType>
        <developerName>BucketField_BusinessSize</developerName>
        <masterLabel>NumericBucket</masterLabel>
        <nullTreatment>z</nullTreatment>
        <sourceColumnName>SALES</sourceColumnName>
        <values>
            <sourceValues>
                <to>10000</to>
            </sourceValues>
            <value>low</value>
        </values>
        <values>
            <sourceValues>
                <from>10000</from>
                <to>25000</to>
            </sourceValues>
            <value>mid</value>
        </values>
        <values>
            <sourceValues>
```

```
            <from>25000</from>
        </sourceValues>
        <value>high</value>
    </values>
</buckets>
<buckets>
    <bucketType>text</bucketType>
    <developerName>BucketField_Region</developerName>
    <masterLabel>TextBucket</masterLabel>
    <nullTreatment>n</nullTreatment>
    <otherBucketLabel>Other</otherBucketLabel>
    <sourceColumnName>ADDRESS1_STATE</sourceColumnName>
    <values>
        <sourceValues>
            <sourceValue>CA</sourceValue>
        </sourceValues>
        <value>west</value>
    </values>
    <values>
        <sourceValues>
            <sourceValue>NY</sourceValue>
        </sourceValues>
        <sourceValues>
            <sourceValue>Ontario</sourceValue>
        </sourceValues>
        <value>east</value>
    </values>
</buckets>
<chart>
    <backgroundColor1>#FFFFFF</backgroundColor1>
    <backgroundColor2>#FFFFFF</backgroundColor2>
    <backgroundFadeDir>Diagonal</backgroundFadeDir>
    <chartSummaries>
        <axisBinding>y</axisBinding>
        <column>FORMULA1</column>
    </chartSummaries>
    <chartSummaries>
        <axisBinding>y</axisBinding>
        <column>FORMULA2</column>
    </chartSummaries>
    <chartSummaries>
        <aggregate>Maximum</aggregate>
        <axisBinding>y</axisBinding>
        <column>CRT_Object__c$Number__c</column>
    </chartSummaries>
    <chartSummaries>
        <axisBinding>y</axisBinding>
        <column>RowCount</column>
    </chartSummaries>
    <chartType>VerticalColumn</chartType>
    <groupingColumn>CRT_Object__c$LastModifiedDate</groupingColumn>
    <legendPosition>Right</legendPosition>
    <location>CHART_TOP</location>
    <size>Medium</size>
```

```
        <summaryAxisRange>Auto</summaryAxisRange>
        <textColor>#000000</textColor>
        <textSize>12</textSize>
        <titleColor>#000000</titleColor>
        <titleSize>18</titleSize>
    </chart>
    <columns>
        <field>CRT_Object__c$Name</field>
    </columns>
    <columns>
        <aggregateTypes>Average</aggregateTypes>
        <field>CRT_Object__c$Currency__c</field>
    </columns>
    <columns>
        <aggregateTypes>Maximum</aggregateTypes>
        <field>CRT_Object__c$Number__c</field>
    </columns>
    <columns>
        <field>BucketField__Region</field>
    </columns>
    <format>Matrix</format>
    <groupingsAcross>
        <dateGranularity>Day</dateGranularity>
        <field>CRT_Object__c$Id</field>
        <sortOrder>Asc</sortOrder>
    </groupingsAcross>
    <groupingsAcross>
        <dateGranularity>Year</dateGranularity>
        <field>CRT_Object__c$LastModifiedDate</field>
        <sortOrder>Asc</sortOrder>
    </groupingsAcross>
    <groupingsDown>
        <dateGranularity>Day</dateGranularity>
        <field>CRT_Object__c$CreatedBy</field>
        <sortOrder>Asc</sortOrder>
    </groupingsDown>
    <groupingsDown>
        <dateGranularity>Day</dateGranularity>
        <field>CRT_Object__c$Currency__c</field>
        <sortOrder>Desc</sortOrder>
    </groupingsDown>
    <name>CrtMMVC</name>
    <reportType>CRT1__c</reportType>
    <scope>organization</scope>
    <showDetails>false</showDetails>
    <timeFrameFilter>
        <dateColumn>CRT_Object__c$CreatedDate</dateColumn>
        <interval>INTERVAL_CUSTOM</interval>
    </timeFrameFilter>
</Report>
```

Declarative Metadata Sample Definition for a Joined Report

A sample XML report definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Report xmlns="http://soap.sforce.com/2006/04/metadata">
<!-- This is a cross-block custom summary formula. Note that the calculated formula reference
 for a blocks reference uses the BlockId#Aggregate. -->
    <aggregates>
        <calculatedFormula>B1#AMOUNT:SUM+B2#EMPLOYEES:SUM</calculatedFormula>
        <datatype>number</datatype>
        <developerName>FORMULA</developerName>
        <isActive>true</isActive>
        <isCrossBlock>true</isCrossBlock>
        <masterLabel>Cross-Block CSF Example</masterLabel>
        <scale>2</scale>
    </aggregates>
<!-- This is a standard custom summary formula. Note that the calculated formula reference
 does not have block reference but just the aggregate name of the report type associated
(Opportunity).-->
    <aggregates>
        <calculatedFormula>AMOUNT:SUM</calculatedFormula>
        <developerName>FORMULA2</developerName>
        <isActive>true</isActive>
        <isCrossBlock>false</isCrossBlock>
        <masterLabel>Standard CSF Example</masterLabel>
        <reportType>Opportunity</reportType>
        <scale>2</scale>
    </aggregates>
    <block>
      <blockInfo>
<!-- This is how the block defines that the custom summary formula should be referenced.
In this example, it's the in standard FORMULA 2 defined above. This block report has blockID
 B1.-->
        <aggregateReferences>
          <aggregate>FORMULA2</aggregate>
        </aggregateReference>
        <blockId>B1</blockId>
        <joinTable>a</joinTable>
      </blockInfo>
      <columns>
        <field>TYPE</field>
      </columns>
      <format>Summary</format>
      <name>Opportunities BLock 3</name>
      <params>
        <name>role_territory</name>
        <value>role</value>
      </params>
      <params>
        <name>terr</name>
        <value>all</value>
      </params>
      <params>
        <name>open</name>
```

```
        <value>all</value>
      </params>
      <params>
        <name>probability</name>
        <value>0</value>
      </params>
      <params>
        <name>co</name>
        <value>1</value>
      </params>
      <reportType>Opportunity</reportType>
      <scope>organization</scope>
      <timeFrameFilter>
        <dateColumn>CLOSE_DATE</dateColumn>
        <interval>INTERVAL_CUSTOM</interval>
      </timeFrameFilter>
    </block>
    <block>
      <blockInfo>
<!-- This is how the block defines that the custom summary formula should be referenced.
In this example, it's the cross-block custom summary formula FORMULA 1 defined above. This
 block report has blockId B2.-->
        <aggregateReferences>
          <aggregate>FORMULA1</aggregate>
        </aggregateReferences>
        <blockId>B2</blockId>
        <joinTable>a</joinTable>
      </blockInfo>
      <columns>
        <field>USERS.NAME</field>
      </columns>
      <columns>
        <field>TYPE</field>
      </columns>
      <columns>
         <field>DUE_DATE</field>
      </columns>
      <columns>
        <field>LAST_UPDATE</field>
      </columns>
      <columns>
        <field>ADDRESS1_STATE</field>
      </columns>
      <format>Summary</format
      <name>Accounts block 5</name>
      <params>
        <name>terr</name>
        <value>all</value>
      </params>
      <params>
        <name>co</name>
        <value>1</value>
      </params>
      <reportType>AccountList</reportType>
```

```
      <scope>organization</scope>
      <timeFrameFilter>
        <dateColumn>CREATED_DATE</dateColumn>
        <interval>INTERVAL_CUSTOM</interval>
      </timeFrameFilter>
    </block>
    <blockInfo>
      <blockId xsi:nil="true"/>
      <joinTable>a</joinTable>
    </blockInfo>
<chart>
        <backgroundColor1>#FFFFFF</backgroundColor1>
        <backgroundColor2>#FFFFFF</backgroundColor2>
        <backgroundFadeDir>Diagonal</backgroundFadeDir>
        <chartSummaries>
            <axisBinding>y</axisBinding>
<!-- This is how chart aggregates are designed in multiblock. We're using RowCount from
Block 1.-->
            <column>B1#RowCount</column>
        </chartSummaries>
        <chartType>HorizontalBar</chartType>
        <enableHoverLabels>false</enableHoverLabels>
        <expandOthers>true</expandOthers>
        <groupingColumn>ACCOUNT_NAME</groupingColumn>
        <location>CHART_TOP</location>
        <showAxisLabels>true</showAxisLabels>
        <showPercentage>false</showPercentage>
        <showTotal>false</showTotal>
        <showValues>false</showValues>
        <size>Medium</size>
        <summaryAxisRange>Auto</summaryAxisRange>
        <textColor>#000000</textColor>
        <textSize>12</textSize>
        <titleColor>#000000</titleColor>
        <titleSize>18</titleSize>
    </chart>
    <format>MultiBlock</format>
    <groupingsDown>
        <dateGranularity>Day</dateGranularity>
        <field>ACCOUNT_NAME</field>
        <sortOrder>Asc</sortOrder>
    </groupingsDown>
    <name>mb_mbapi</name>
    <reportType>Opportunity</reportType>
    <showDetails>true</showDetails>
</Report>
```

SEE ALSO:

Dashboard

ReportType

Represents the metadata associated with a custom report type. This type extends the Metadata metadata type and inherits its `fullName` field. Custom report types allow you to build a framework from which users can create and customize reports. For more information, see "Set Up a Custom Report Type" in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.reportType` for the custom report type definition. There is one file per custom report type. Report types are stored in the `reportTypes` directory of the corresponding package directory.

Version

Custom report types are available in API version 14.0 and later.

Fields

Field Name	Field Type	Description
`autogenerated`	boolean	Indicates that the report type was automatically generated when historical trending was enabled for an entity. Available in API version 29 and later.
`baseObject`	string	Required. The primary object for the custom report type, for example, Account. All objects, including custom objects, are supported. You cannot edit this field after initial creation.
`category`	ReportTypeCategory (enumeration of type string)	Required. This field controls the category for the report. The valid values are: • `accounts` • `opportunities` • `forecasts` • `cases` • `leads` • `campaigns` • `activities` • `busop` • `products` • `admin` • `territory` • `territory2` (This value is available in API version 31.0 and later.) • `usage_entitlement` • `wdc` (This value is available in API version 29.0 and later.) • `calibration` (This value is available in API version 29.0 and later.)

Field Name	Field Type	Description
		• `other` • `content`
`deployed`	boolean	Required. Indicates whether the report type is available to users (`true`) or whether it's still in development (`false`).
`description`	string	The description of the custom report type.
`fullName`	string	The report type developer name used as a 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 end with an underscore, and not contain two consecutive underscores.
`join`	ObjectRelationship	The object joined to the `baseObject`. For example, Contacts may be joined to the primary Accounts object.
`label`	string	Required. The report type label.
`sections`	ReportLayoutSection[]	The groups of columns available for the report type. Though columns are not strictly required, a report without columns is not very useful.

ObjectRelationship

ObjectRelationship represents a join to another object. For more information, see "Add Child Objects To Your Custom Report Type" in the Salesforce online help.

Field Name	Field Type	Description
`join`	ObjectRelationship	This field is a recursive reference that allows you to join more than two objects. A maximum of four objects can be joined in a custom report type. When more than two objects are joined, an inner join is not allowed if there has been an outer join earlier in the join sequence. The `baseObject` is first joined to the object specified in `relationship`; the resulting data set is then joined with any objects specified in this field.
`outerJoin`	boolean	Required. Indicates whether this is an outer join (`true`) or not (`false`). An outer join returns a row even if the joined table does not contain a matching value in the join column.
`relationship`	string	Required. The object joined to the primary object; for example, Contacts.

ReportLayoutSection

ReportLayoutSection represents a group of columns used in the custom report type.

Field Name	Field Type	Description
`columns`	ReportTypeColumn[]	The list of columns projected from the query, defined by this custom report type.

Field Name	Field Type	Description
masterLabel	string	Required. The label for this group of columns in the report wizard.

ReportTypeColumn

ReportTypeColumn represents a column in the custom report type.

Field Name	Field Type	Description
checkedByDefault	boolean	Required. Indicates whether this column is selected be default (true) or not (false).
displayNameOverride	string	A customized column name, if desired.
field	string	Required. The field name associated with the report column.
table	string	Required. The table associated with the field; for example, Account.

Declarative Metadata Sample Definition

The definition of a custom report type is shown below. Account is joined to Contacts and the resulting data set is joined with Assets.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReportType xmlns="http://soap.sforce.com/2006/04/metadata">
    <baseObject>Account</baseObject>
    <category>accounts</category>
    <deployed>true</deployed>
    <description>Account linked to Contacts and Assets</description>
    <join>
        <join>
            <outerJoin>false</outerJoin>
            <relationship>Assets</relationship>
        </join>
        <outerJoin>false</outerJoin>
        <relationship>Contacts</relationship>
    </join>
    <label>Account Contacts and Assets</label>
    <sections>
        <columns>
            <checkedByDefault>true</checkedByDefault>
            <field>obj_lookup__c.Id</field>
            <table>Account</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>obj_lookup__c.Name</field>
            <table>Account</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
```

```
            <field>Opportunity__c.Amount</field>
            <table>Account</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>Owner.IsActive</field>
            <table>Account</table>
        </columns>
        <masterLabel>Accounts</masterLabel>
    </sections>
    <sections>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>Owner.Email</field>
            <table>Account.Contacts</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>byr__c</field>
            <table>Account.Contacts</table>
        </columns>
        <columns>
            <checkedByDefault>true</checkedByDefault>
            <field>ReportsTo.CreatedBy.Contact.Owner.MobilePhone</field>
            <table>Account.Contacts</table>
        </columns>
        <masterLabel>Contacts</masterLabel>
    </sections>
</ReportType>
```

Usage

The custom report type refers to fields by using their API names. For a historical field (one that has `trackTrending` set to `true`) the API name includes `hst`, such as `Field2__c_hst`.

```
<sections>
    <columns>
        <checkedByDefault>false</checkedByDefault>
        <field>Field2__c_hst</field>
        <table>CustomTrendedObject__c.CustomTrendedObject__c_hst</table>
    </columns>
    <masterLabel>History</masterLabel>
</sections>
```

For more information, see `trackTrending` on page 231.

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

This metadata type extends to subtype RoleOrTerritory on page 496.

Field Name	Field Type	Description
`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 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>
```

RoleOrTerritory

Represents the common base type and valid values for role or territory.

Version

RoleOrTerritory components are available in API version 24.0 and later.

Note: You can't create a RoleOrTerritory component directly. Use the Role or Territory metadata types instead.

Fields

Field Name	Field Type	Description
caseAccessLevel	string	Specifies whether a user can access other users' cases that are associated with accounts the user owns. Valid values are: • Read • Edit • None This field is not visible if your organization's sharing model for cases is Public Read/Write. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup.
contactAccessLevel	string	Specifies whether a user can access other users' contacts that are associated with accounts the user owns. Valid values are: • Read • Edit • None This field is not visible if your organization's sharing model for contacts is Public Read/Write or Controlled by Parent. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup.
description	string	The description of the role or territory.
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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
mayForecastManagerShare	boolean	Indicates whether the forecast manager can manually share their own forecast.
name	string	Required. The name of the role or territory.
opportunityAccessLevel	string	Specifies whether a user can access other users' opportunities that are associated with accounts the user owns. Valid values are: • Read • Edit • None This field is not visible if your organization's sharing model for opportunities is Public Read/Write. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup.

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>
```

The following is the definition of a territory.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory xmlns="http://soap.sforce.com/2006/04/metadata">
    <accountAccessLevel>Edit</accountAccessLevel>
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <description>Sample Territory</description>
    <mayForecastManagerShare>false</mayForecastManagerShare>
    <name>T22name</name>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
</Territory>
```

SEE ALSO:

- Role
- Territory

SamlSsoConfig

Represents a SAML Single Sign-On configuration. This type 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 Salesforce.

File Suffix and Directory Location

SamlSsoConfig components have the suffix `.samlssoconfig` and are stored in the `samlssoconfigs` 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 identityLocation is set to Attribute. Possible values include unspecified, emailAddress or persistent. All legal values can be found in the "Name Identifier Format Identifiers" section of the Assertions and Protocols SAML 2.0 specification.
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, such as a public site Visualforce page. The URL can be absolute or relative.
executionUserId	string	The user that runs the Apex handler class. The user must have the "Manage Users" permission. A user is required if you specify a SAML JIT handler class.
identityLocation	SamlIdentityLocationType (enumeration of type string)	The location in the assertion where a user should be identified. Valid values are: • 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: • Username — The user's Salesforce 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 Salesforce organization.
issuer	string	The identification string for the Identity Provider.
loginUrl	string	For SAML 2.0 only: The URL where Salesforce 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

Field Name	Field Type	Description
		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 Salesforce 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: • `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.
`requestSignatureMethod`	string	The method that's used to sign the SAML request. Valid values are `RSA-SHA1` and `RSA-SHA256`.
`salesforceLoginUrl`	string	The URL associated with login for the Web single sign-on flow.
`samlEntityId`	string	The issuer in SAML requests generated by Salesforce, and is also the expected audience of any inbound SAML Responses. If you don't have domains deployed, this value is always `https://saml.salesforce.com`. If you have domains deployed, Salesforce recommends that you use your custom domain name.
`samlJitHandlerId`	string	The name of an existing Apex class that implements the `Auth.SamlJitHandler` interface.
`samlVersion`	SamlType (enumeration of type string)	The SAML version in use. Valid values are: • `SAML1_1` — SAML 1.1 • `SAML2_0` — SAML 2.0
`userProvisioning`	boolean	If `true`, Just-in-Time user provisioning is enabled, which creates users on the fly the first time they try to log in. Specify `Federation ID` for the `identityMapping` value to use this feature.
`validationCert`	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>
  <requestSignatureMethod>RSA-SHA1</requestSignatureMethod>
  <salesforceLoginUrl>
    https://login.salesforce.com?so=00DD0000000JxeI
  </salesforceLoginUrl>
  <samlEntityId>
    https://saml.salesforce.com/customPath
  </samlEntityId>
  <samlVersion>SAML2_0</samlVersion>
  <userProvisioning>false</userProvisioning>
  <validationCert>
    MIIEojCCA4qgAwIBAgIOATtxsoBFAAAAAD4...
  </validationCert>
</SamlSsoConfig>
```

Scontrol

Important: Visualforce pages supersede s-controls. Organizations that haven't previously used s-controls can't create them. Existing s-controls are unaffected, and can still be edited.

Deprecated. Represents an Scontrol component, corresponding to an s-control in the Salesforce user interface. For more information, see "About S-Controls" in the Salesforce online help. This metadata type extends the MetadataWithContent component and shares its fields.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.scf` for the s-control file. The accompanying metadata file is named *ScontrolName*`-meta.xml`.

Scontrol components are stored in the `scontrols` folder in the corresponding package directory.

Version

Scontrols are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`content`	base64Binary	Content of the s-control. 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

Field Name	Field Type	Description
		for you by a SOAP client. This field is inherited from the MetadataWithContent component.
`contentSource`	SControlContentSource (enumeration of type string)	Required. Determines how you plan to use the s-control: • `HTML`: Select this option if you want to enter the content for your s-control in `content`. • `URL`: Select this option if you want to enter the link or URL of an external website in `content`. • `Snippet`: Snippets are s-controls that are designed to be included in other s-controls. Select this option if you want to enter the content for your s-control snippet in `content`.
`description`	string	Optional text that describes the s-control. This only displays to users with "View All Data" permission (administrator).
`encodingKey`	Encoding (enumeration of type string)	Required. The default encoding setting is Unicode: `UTF-8`. Change it if you are passing information to a URL that requires data in a different format. This option is available when you select `URL` as the value for `contentSource`.
`fileContent`	base64	File contents displayed if you add this s-control to a custom link. The file can contain a Java applet, Active-X control, or any other type of content you want. This option only applies to s-controls with a value of `HTML` for `contentSource`.
`fileName`	string	The unique name for the s-control. This name can contain only underscores and alphanumeric characters, and must be unique in your organization. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field cannot be changed for components installed by a managed package. It is only relevant if the `fileContent` field also has a value. This is a new field in API version 14.0.
`fullName`	string	The s-control developer name used as a 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 end with an underscore, and not contain two consecutive underscores. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the `name` field. This field is inherited from the Metadata component.
`name`	string	Required. The unique name for the s-control. It must contain alphanumeric characters only and begin with a letter. For example `example_s_control`.
`supportsCaching`	boolean	Required. Indicates whether the s-control supports caching (`true`) or not (`false`). Caching optimizes the page so that it remembers

Field Name	Field Type	Description
		which s-controls are on the page when it reloads. This option only applies to HTML s-controls.

Declarative Metadata Sample Definition

The following sample creates the `Myriad_Publishing.scf` s-control, which creates a link to the website specified in the s-control. The corresponding `Myriad_Publishing.scf-meta.xml` metadata file follows the s-control file.

`Myriad_Publishing.scf` file:

```
http://www.myriadpubs.com
```

`Myriad_Publishing.scf-meta.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Scontrol xmlns="http://soap.sforce.com/2006/04/metadata">
    <contentSource>URL</contentSource>
    <description>s-control to open Myriad Publishing website.</description>
    <encodingKey>UTF-8</encodingKey>
    <name>Myriad Publishing</name>
    <supportsCaching>true</supportsCaching>
</Scontrol>
```

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 113 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
- AddressSettings
- CaseSettings
- ChatterAnswersSettings
- CompanySettings
- ContractSettings
- EntitlementSettings
- ForecastingSettings
- IdeasSettings
- KnowledgeSettings
- MobileSettings
- SecuritySettings

AccountSettings

Represents an organization's account settings for account teams, account owner report, and the **View Hierarchy** link. This type 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 because there is only one settings file for each settings component.

Version

AccountSettings is available in API versions 29.0 and later.

Fields

Field Name	Field Type	Description
`enableAccountOwnerReport`	boolean	Indicates whether Account Owner Report may (`true`) or may not (`false`) be run by all users.
`enableAccountTeams`	boolean	Indicates whether Account Teams are enabled (`true`) or not (`false`). The Metadata API cant' be used to disable Account Teams.
`showViewHierarchyLink`	boolean	Indicates whether the default **View Hierarchy** link on all business account detail pages is visible (`true`) or hidden (`false`).

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 because 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
allowUsersToRelateMultipleContactsToTasksAndEvents	boolean	This read only field indicates whether Shared Activities is enabled. When the value is true, allows users to relate multiple contacts to a task or event. **Important:** Beginning with API v36.0, this field is read-only in all versions of the API. You can't change the value of this field. Even though this field was updateable before Spring '16, changing this field's value wasn't supported and could have resulted in an incorrect integration. If you have code in older API versions that changes the value of this field, ensure you update that code to prevent any errors.
`enableActivityReminders`	boolean	Enables popup activity reminders for an organization. Administrators control this field on the Activity settings page.

Field Name	Field Type	Description
enableClickCreateEvents	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.
enableDragAndDropScheduling	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.
enableEmailTracking	boolean	Enables tracking of outbound HTML emails if an organization uses HTML email templates. Administrators control this field on the Activity settings page.
enableGroupTasks	boolean	Lets users assign independent copies of a new task to multiple users. Administrators control this field on the Activity settings page.
enableListViewScheduling	boolean	Extends the functionality of enableDragAndDropScheduling and enableClickCreateEvents to list view calendars. Administrators control this field on the User Interface settings page.
enableMultidayEvents	boolean	Enables creation of events that end more than 24 hours after they start. Administrators control this field on the Activity settings page.
enableRecurringEvents	boolean	Enables creation of events that repeat at specified intervals. Administrators control this field on the Activity settings page.
enableRecurringTasks	boolean	Enables creation of tasks that repeat at specified intervals. Administrators control this field on the Activity settings page.
enableSidebarCalendarShortcut	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.
enableSimpleTaskCreateUI	boolean	Allows administrators to specify whether tapping New Task in Salesforce1 opens a regular task record edit page or a page that displays key task fields first. Administrators control this field on the Activity settings page.

Field Name	Field Type	Description
enableUNSTaskDelegatedToNotifications	boolean	On the Activity settings page, exposes a setting for administrators to hide or show a user setting that lets individual users enable or disable email notifications when tasks are assigned to them.
meetingRequestsLogo	string	Available when showCustomLogoMeetingRequests 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.
showCustomLogoMeetingRequests	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.
showEventDetailsMultiUserCalendar	boolean	Displays event details on-screen rather than in hover text. Administrators control this field on the Activity settings page.
showHomePageHoverLinksForEvents	boolean	In the calendar section of the Home tab: • 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.
showMyTasksHoverLinks	boolean	In the My Tasks section of the Home tab and on the calendar day view: • 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.
showRequestedMeetingsOnHomePage	boolean	In the Calendar on the Home tab, displays the Requested Meetings subtab, listing the meetings a user has requested but not confirmed. 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">
    <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>
```

SEE ALSO:

Document

AddressSettings

Represents the configuration of country and state picklists. Use the AddressSettings component type to configure state and country data in your organization so that you can convert text-based values into standard picklist values. To convert your state and country values, from Setup, enter *`State and Country Picklists`* in the `Quick Find` box, then select **State and Country Picklists** . For more information, see "State and Country Picklists" in the Salesforce online help.

This type 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.

Declarative Metadata File Suffix and Directory Location

AddressSettings values are stored in a single file named `Address.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

AddressSettings is available in API versions 27.0 and later.

CountriesAndStates

This complex metadata type represents valid definitions of states and countries in picklists.

Note: You can use the Metadata API to edit existing states and countries in state and country picklists. You can't use the Metadata API to create or delete new states or countries.

Field	Field Type	Description
`countries`	Country[]	The countries available in picklists.

Country

This metadata type provides the definition for a country in a picklist.

Field	Field Type	Description
`active`	boolean	Determines whether the value is available in the API. Important: After you enable state and country picklists in your Salesforce organization, you can't set the `active` status to *`false`*.
`integrationValue`	string	A customizable text value that is linked to a state or country code. Integration values for standard states and countries default to the full ISO-standard state and country names. Integration values function similarly to the API names of custom fields and objects. Configuring integration values allows integrations that you set up before enabling state and country picklists to continue to work. Important: If you don't specify integration values before enabling state and country picklists in your organization, records use the default value provided by Salesforce. If you change integration values later, records created or updated from that point on use your edited values.
`isoCode`	string	The ISO-standard code populates this field when you issue a `retrieve()` call. This field is read only in the API but you can edit the label in Setup. You can't edit the `isoCode` of `standard` states and countries.

Field	Field Type	Description
`label`	string	The label is what users see in picklists in Salesforce. This field is read only in the API but you can edit the label in Setup.
`orgDefault`	boolean	Sets a country as the default value for new records in the Salesforce organization.
`standard`	boolean	Standard states and countries are states and countries that are included with Salesforce. You can't edit the `standard` attribute.
`states`	State[]	The states or provinces that are part of the country.
`visible`	boolean	Makes the state or country available to users in Salesforce. States or countries that are `visible` must also be `active`.

State

This metadata type provides the definition for a state in a picklist.

Field	Field Type	Description
`active`	boolean	Determines whether the value is available in the API. **Important:** After you enable state and country picklists in your Salesforce organization, you can't set the `active` status to *`false`*.
`integrationValue`	string	A customizable text value that is linked to a state or country code. Integration values for standard states and countries default to the full ISO-standard state and country names. Integration values function similarly to the API names of custom fields and objects. Configuring integration values allows integrations that you set up before enabling state and country picklists to continue to work. **Important:** If you don't specify integration values before enabling state and country picklists in your organization, records use the default value provided by Salesforce. If you change integration values later, records created or updated from that point on use your edited values.
`isoCode`	string	The ISO-standard code populates this field when you issue a `retrieve()` call. This field is read only in the API but you can edit the label in Setup.
`label`	string	The label is what users see in picklists in Salesforce. This field is read only in the API but you can edit the label in Setup.

Field	Field Type	Description
`standard`	boolean	Standard states and countries are states and countries that are included with Salesforce. You can't edit the `standard` attribute.
`visible`	boolean	Makes the state or country available to users in Salesforce. States or countries that are `visible` must also be `active`.

Declarative Metadata Sample Definition

The following is sample XML that configures state and country picklists for the United States and Canada for use in an organization. It also makes the country of Greenland available only in the API. This example is supported in API version 36.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<AddressSettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <countriesAndStates>
    <countries>
      <country>
        <active>true</active>
        <integrationValue>United States</integrationValue>
        <isoCode>US</isoCode>
        <label>United States</label>
        <orgDefault>true</orgDefault>
        <standard>true</standard>
        <states>
          <state>
            <active>true</active>
            <integrationValue>Alabama</integrationValue>
            <isoCode>AL</isoCode>
            <label>Alabama</label>
            <standard>true</standard>
            <visible>true</visible>
          </state>
          <state>
            <active>true</active>
            <integrationValue>Alaska</integrationValue>
            <isoCode>AK</isoCode>
            <label>Alaska</label>
            <standard>true</standard>
            <visible>true</visible>
          </state>
        </states>
        <visible>true</visible>
      </country>
      <country>
        <active>true</active>
        <integrationValue>Canada</integrationValue>
        <isoCode>CA</isoCode>
        <label>Canada</label>
        <orgDefault>false</orgDefault>
        <states>
          <state>
```

```
            <active>true</active>
            <integrationValue>Alberta</integrationValue>
            <isoCode>AB</isoCode>
            <label>Alberta</label>
             <standard>true</standard>
             <visible>true</visible>
          </state>
          <state>
            <active>true</active>
            <integrationValue>British Columbia</integrationValue>
            <isoCode>BC</isoCode>
            <label>British Columbia</label>
            <standard>true</standard>
            <visible>true</visible>
          </state>
        </states>
        <visible>true</visible>
      </country>
      <country>
        <active>true</active>
        <integrationValue>Greenland</integrationValue>
        <isoCode>GL</isoCode>
        <label>Greenland</label>
        <standard>true</standard>
        <visible>false</visible>
      </country>
    </countries>
  </countriesAndStates>
</AddressSettings>
```

SEE ALSO:

Settings

BusinessHoursSettings

Represents the metadata used to manage settings for business hours and holidays in entitlements, entitlement templates, campaigns, and cases. This type 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 because 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.

Field Name	Field Type	Description
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 recurenceInstance value is First, the holiday recurs on the first Monday of the month every 3 months.
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>
```

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>
```

CaseSettings

Represents an organization's case settings, such as the default case owner, which case-related features are enabled, and which email templates are used for various case activities.

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

CaseSettings values are stored in the `Case.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

CaseSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
caseAssignNotificationTemplate	string	Specifies the email template used for case assignment notifications. The format must be folderName/emailTemplateName.
caseCloseNotificationTemplate	string	Specifies the email template used for case close notifications.The format must be folderName/emailTemplateName.
caseCommentNotificationTemplate	string	Specifies the email template used for case comment notifications. The format must be folderName/emailTemplateName.
caseCreateNotificationTemplate	string	Specifies the email template used for case create notifications. The format must be folderName/emailTemplateName.
caseFeedItemSettings	FeedItemSettings[]	Specifies the settings for feed items in feed-based case page layouts. This field is available in API version 32.0 and later.
closeCaseThroughStatusChange	boolean	Indicates whether Closed is included in the Case Status field on case edit pages (true) or not (false).
defaultCaseOwner	string	Specifies the default owner of a case when assignment rules fail to locate an owner.
defaultCaseOwnerType	string	Specifies whether the default case owner is a user or a queue.
defaultCaseUser	string	Specifies the user listed in the Case History related list for automated case changes from: • Assignment rules • Escalation rules • On-Demand Email-to-Case • Cases logged in the Self-Service portal
emailToCase	EmailToCaseSettings	The organization's Email-to-Case settings.
enableCaseFeed	boolean	Indicates whether Case Feed is enabled (true) or not (false).
enableDraftEmails	boolean	Indicates whether draft emails are enabled (true) or not (false). Enabling email drafts requires that Case Feed and Email-to-Case are also enabled.
enableEarlyEscalationRuleTriggers	boolean	Indicates whether early triggers on escalation rules are enabled (true) or not (false).

Field Name	Field Type	Description
enableNewEmailDefaultTemplate	boolean	Indicates whether default email templates are enabled (true) or not (false). Default email templates are available only if draft emails are enabled.
enableSuggestedArticlesApplication	boolean	Indicates whether the Suggested Articles list appears on case pages.(true) or not (false). Is only valid if enableSuggestedSolutions=false.
enableSuggestedArticlesCustomerPortal	boolean	Indicates whether the Suggested Articles list appears on customer portal pages (true) or not (false). Is only valid if enableSuggestedSolutions=false.
enableSuggestedArticlesPartnerPortal	boolean	Indicates whether the Suggested Articles list appears on partner portal pages (true) or not (false). Is only valid if enableSuggestedSolutions=false.
enableSuggestedSolutions	boolean	Indicates whether the View Suggested Solutions or Find Articles button appears on case detail pages (true) or not (false). Is only valid if enableSuggestedArticlesApplication, enableSuggestedArticlesCustomerPortal, and enableSuggestedArticlesPartnerPortal=false.
keepRecordTypeOnAssignmentRule	boolean	Indicates whether, when applying assignment rules to manually created records, to keep the existing record type (true) or to override the existing record type with the assignee's default record type (false).
newEmailDefaultTemplateClass	string	Specifies the Apex class that defines the default email template for new email messages in Case Feed. This field appears only when enableNewEmailDefaultTemplate=true.
notifyContactOnCaseComment	boolean	Indicates whether contacts who are not members of your Self-Service portal can be notified when a new comment is added to a case.(true) or not (false).
notifyDefaultCaseOwner	boolean	Indicates whether the default case owner is notified when assigned a new case (true) or not (false).
notifyOwnerOnCaseComment	boolean	Indicates whether the case owner is notified when a comment is added to a case (true) or not (false).
notifyOwnerOnCaseOwnerChange	boolean	Indicates whether the Send Notification Email checkbox on cases is automatically selected when users change a case owner to another user (true).
showFewerCloseActions	boolean	Indicates whether the **Save & Close** button on case edit pages and the **Cls** link on Cases related lists are hidden (true) or shown (false).

Field Name	Field Type	Description
useSystemEmailAddress	boolean	Indicates whether case comment, case attachment, and case assignment email notifications are sent from a system address (true) or whether case notifications appear to be sent from the user or contact updating the case (false).
webToCase	WebToCaseSettings	The organization's Web-to-Case settings.

EmailToCaseSettings

Represents an organization's Email-to-Case settings.

Fields

Field Name	Field Type	Description
enableEmailToCase	boolean	Indicates whether Email-to-Case is enabled (true) or not (false). Note: once Email-to-Case is enabled, it can't be disabled.
enableHtmlEmail	boolean	Indicates whether HTML email is enabled (true) or not (false).
enableOnDemandEmailToCase	boolean	Indicates whether On-Demand Email-to-Case is enabled (true) or not (false).
enableThreadIDInBody	boolean	Indicates whether the Thread ID for a case is inserted in the body of an email (true) or not (false).
enableThreadIDInSubject	boolean	Indicates whether the Thread ID for a case is inserted in the subject line of an email (true) or not (false).
notifyOwnerOnNewCaseEmail	boolean	Indicates whether the owner of a case receives a notification when a new email related to the case is received (true) or not (false).
overEmailLimitAction	EmailToCaseOnFailureActionType (enumeration of type string)	Specifies what happens to email messages received after an organization exceeds its daily Email-to-Case limits. Valid values are: • Bounce • Discard • Requeue
preQuoteSignature	boolean	Indicates whether the user signature is inserted after the reply but before the email thread in an outbound email (true) or at the end of the email (false).
routingAddresses	EmailToCaseRoutingAddress[]	The organization's Email-to-Case routing address settings.

Field Name	Field Type	Description
unauthorizedSenderAction	EmailToCaseOnFailureActionType (enumeration of type string)	Specifies what happens to email messages received from invalid senders. Valid values are: • Bounce • Discard

EmailToCaseRoutingAddress

Represents an organization's Email-to-Case routing address.

Fields

Field Name	Field Type	Description
addressType	EmailToCaseRoutingAddressType (enumeration of type string)	Specifies the type of Email-to-Case routing address. Valid values are: • EmailToCase—A routing address used with Email-to-Case or On-Demand Email-to-Case. • Outlook—A routing address used with Salesforce for Outlook to create cases from Outlook. Requires that On-Demand Email-to-Case is enabled.
authorizedSenders	string	Specifies the email addresses or domains from which On-Demand Email-to-Case can receive email. Include multiple entries in a comma-separated list.
caseOrigin	string	Specifies the default case origin for cases created through this routing address.
caseOwner	string	Specifies the default owner of cases created through this routing address. The owner is specified by Salesforce username.
caseOwnerType	string	Specifies whether the default case owner is a user or a queue.
casePriority	string	Specifies the default case priority for cases created through this routing address.
createTask	boolean	Indicates whether a task is automatically assigned to the case owner when a case is created through an email (true) or not (false).
emailAddress	string	Specifies the email address used to route email messages that are submitted as cases.
routingName	string	Specifies the name of the Email-to-Case routing address.
saveEmailHeaders	boolean	Indicates whether email routing and envelope information is saved (true) or not (false).

Field Name	Field Type	Description
`taskStatus`	string	Specifies the default status on tasks automatically assigned to the case owner when email is submitted as a case. Only applies if `createTask` is set to `true`.

FeedItemSettings

Represents an organization's feed item settings. Available in API version 32.0 and later.

Field Name	Field Type	Description
`characterLimit`	int	Specifies the maximum number of characters displayed for each feed item.
`collapseThread`	boolean	Indicates whether earlier messages in an email thread are removed from email feed items(`true`) or not (`false`).
`displayFormat`	FeedItemDisplayFormat (enumeration of type string)	Indicates how email feed items are displayed. Valid values are: • `Default`—Blank lines in email feed items are displayed. • `HideBlankLines`—Blank lines in email feed items are not displayed.
`feedItemType`	FeedItemType (enumeration of type string)	The type of feed item to which the settings apply. For `FeedItemSettings`, the only valid `feedItemType` value is `EmailMessageEvent`.

WebToCaseSettings

Represents an organization's Web-to-Case settings.

Fields

Field Name	Field Type	Description
`caseOrigin`	string	Specifies the default case origin for cases created through this web form. Only applies if `enableWebToCase` is set to `true`.
`defaultResponseTemplate`	string	Specifies the default template used for email responses to cases submitted through a Self-Service portal. Only applies if `enableWebToCase` is set to `true`.
`enableWebToCase`	boolean	Indicates whether Web-to-Case is enabled (`true`) or not (`false`).

Declarative Metadata Sample Definition

This is a sample case settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<CaseSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseAssignNotificationTemplate>
        unfiled$public/SupportCaseAssignmentNotification
    </caseAssignNotificationTemplate>
    <caseCloseNotificationTemplate>
        unfiled$public/SupportCaseCloseNotification
    </caseCloseNotificationTemplate>
    <caseCommentNotificationTemplate>
        unfiled$public/SupportCaseCommentNotification
    </caseCommentNotificationTemplate>
    <caseCreateNotificationTemplate>
        unfiled$public/SupportCaseCreateNotification
    </caseCreateNotificationTemplate>
    <closeCaseThroughStatusChange>true</closeCaseThroughStatusChange>
    <defaultCaseOwner>admin@acme.com</defaultCaseOwner>
    <defaultCaseOwnerType>User</defaultCaseOwnerType>
    <defaultCaseUser>admin@acme.com</defaultCaseUser>
    <emailToCase>
        <enableEmailToCase>true</enableEmailToCase>
        <enableHtmlEmail>false</enableHtmlEmail>
        <enableOnDemandEmailToCase>true</enableOnDemandEmailToCase>
        <enableThreadIDInBody>true</enableThreadIDInBody>
        <enableThreadIDInSubject>true</enableThreadIDInSubject>
        <notifyOwnerOnNewCaseEmail>false</notifyOwnerOnNewCaseEmail>
        <overEmailLimitAction>Bounce</overEmailLimitAction>
        <preQuoteSignature>true</preQuoteSignature>
        <routingAddresses>
            <addressType>EmailToCase</addressType>
            <authorizedSenders>user@acme.com</authorizedSenders>
            <caseOrigin>Email</caseOrigin>
            <casePriority>Medium</casePriority>
            <createTask>true</createTask>
            <emailAddress>support@acme.com</emailAddress>
            <routingName>EmailToCaseRoutingAddress1</routingName>
            <saveEmailHeaders>true</saveEmailHeaders>
            <taskStatus>Not Started</taskStatus>
        </routingAddresses>
        <routingAddresses>
            <addressType>Outlook</addressType>
            <authorizedSenders>user@acme.com</authorizedSenders>
            <caseOrigin>Email</caseOrigin>
            <caseOwner>admin@acme.com</caseOwner>
            <caseOwnerType>User</caseOwnerType>
            <casePriority>High</casePriority>
            <routingName>OutlookRoutingAddress1</routingName>
        </routingAddresses>
        <unauthorizedSenderAction>Discard</unauthorizedSenderAction>
    </emailToCase>
    <enableCaseFeed>true</enableCaseFeed>
    <enableDraftEmails>true</enableDraftEmails>
```

```
    <enableEarlyEscalationRuleTriggers>true</enableEarlyEscalationRuleTriggers>
    <enableNewEmailDefaultTemplate>true</enableNewEmailDefaultTemplate>
    <enableSuggestedArticlesApplication>true</enableSuggestedArticlesApplication>
    <enableSuggestedArticlesCustomerPortal>true</enableSuggestedArticlesCustomerPortal>
    <enableSuggestedArticlesPartnerPortal>false</enableSuggestedArticlesPartnerPortal>
    <enableSuggestedSolutions>false</enableSuggestedSolutions>
    <keepRecordTypeOnAssignmentRule>true</keepRecordTypeOnAssignmentRule>
    <newEmailDefaultTemplateClass>CaseTemplateController</newEmailDefaultTemplateClass>
    <notifyContactOnCaseComment>true</notifyContactOnCaseComment>
    <notifyDefaultCaseOwner>true</notifyDefaultCaseOwner>
    <notifyOwnerOnCaseComment>true</notifyOwnerOnCaseComment>
    <notifyOwnerOnCaseOwnerChange>false</notifyOwnerOnCaseOwnerChange>
    <showFewerCloseActions>false</showFewerCloseActions>
    <useSystemEmailAddress>true</useSystemEmailAddress>
    <webToCase>
        <caseOrigin>Web</caseOrigin>
       <defaultResponseTemplate>unfiled$public/SupportCaseResponse</defaultResponseTemplate>

        <enableWebToCase>true</enableWebToCase>
    </webToCase>
</CaseSettings>
```

SEE ALSO:

Settings

ChatterAnswersSettings

Represents the metadata used to manage settings for Chatter Answers.

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

Chatter Answers settings are stored in a single file named `ChatterAnswers.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ChatterAnswersSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`emailFollowersOnBestAnswer`	boolean	Indicates whether users are notified when a best answer is selected for a question that they’re following (`true`) or not (`false`).
`emailFollowersOnReply`	boolean	Indicates whether users are notified when other users reply to questions they’re following (`true`) or not (`false`).

Field Name	Field Type	Description
`emailOwnerOnPrivateReply`	boolean	Indicates whether users are notified when customer support responds to their questions privately (`true`) or not (`false`).
`emailOwnerOnReply`	boolean	Indicates whether users are notified when other users reply to their questions (`true`) or not (`false`).
`enableAnswerViaEmail`	boolean	Indicates whether users can post answers by replying to email notifications (`true`) or not (`false`). This field is available in API version 29.0 and later.
`enableChatterAnswers`	boolean	Indicates whether Chatter Answers is enabled in the organization (`true`) or not (`false`).
`enableFacebookSSO`	boolean	Indicates whether users sign in to your Chatter Answers communities with their Facebook logins (`true`) or not (`false`). To enable this feature, you must define and enable a Facebook authentication provider in your organization's security controls and enable Auth Providers in your organization.
`enableInlinePublisher`	boolean	Indicates whether users can filter search results by articles or questions before they post a question to any of your Chatter Answers communities (`true`) or not (`false`). Also, adds `Title` and `Body` fields to questions for easier text input and scanning. This field is available in API version 29.0 and later.
`enableReputation`	boolean	Indicates whether reputations display for users as hover text on their profile pictures (`true`) or not (`false`). Reputation is enabled across all zones. To enable the reputation setting, you must enable Reputation in your organization.
`enableRichTextEditor`	boolean	Indicates whether the rich text editor is enabled for users to format text and upload images when posting questions (`true`) or not (`false`). To enable rich text editor, you must enable Optimize Question Flow.
`facebookAuthProvider`	string	The name of an existing Facebook authentication provider. You must choose a Facebook authentication provider to implement Facebook Single Sign On for your Chatter Answers communities.
`showInPortals`	boolean	Indicates whether Chatter Answers can be added as a tab to your Customer portal or partner portal (`true`) or not (`false`).

Declarative Metadata Sample Definition

The following is an example `chatteranswers.settings` metadata file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ChatterAnswersSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <emailFollowersOnBestAnswer>true</emailFollowersOnBestAnswer>
    <emailFollowersOnReply>true</emailFollowersOnReply>
    <emailOwnerOnPrivateReply>true</emailOwnerOnPrivateReply>
    <emailOwnerOnReply>true</emailOwnerOnReply>
```

```
    <enableChatterAnswers>true</enableChatterAnswers>
    <enableFacebookSSO>true</enableFacebookSSO>
    <enableInlinePublisher>true</enableInlinePublisher>
    <enableReputation>true</enableReputation>
    <enableRichTextEditor>true</enableRichTextEditor>
    <facebookAuthProvider>FacebookAuthProvider</facebookAuthProvider>
    <showInPortals>true</showInPortals>
</ChatterAnswersSettings>
```

The following is an example `package.xml` manifest that references the ChatterAnswersSettings definitions:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ChatterAnswers</members>
        <name>Settings</name>
    </types>
    <version>29.0</version>
</Package>
```

SEE ALSO:

Settings

CompanySettings

Represents global settings that affect multiple features in your organization.

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

CompanySettings values are stored in a single file named `Company.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Company Profile Settings are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`fiscalYear`	FiscalYearSetting	The organization's fiscal year setting based on year and start month. Not available if Custom Fiscal Year or Forecasts (Classic) is enabled. When changing fiscal year settings, quotas and adjustments can be purged. For example changing your start month results in purging this data.

FiscalYearSetting

Represents your organization's fiscal year setting.

Field	Field Type	Description
`fiscalYearNameBasedOn`	string	This field is used to determine the fiscal year name. Valid values are `endingMonth` or `startingMonth`. For example, if your fiscal year starts in April 2012 and ends in March 2013, and this value is: • `endingMonth`, then 2013 is used for the fiscal year name. • `startingMonth`, then 2012 is used for the fiscal year name.
`startMonth`	string	The month on which the fiscal year is based.

Declarative Metadata Sample Definition — Fiscal Year Setting

A sample XML definition of a fiscal year setting is shown below. Note that this example is supported in API version 27.0 and later.

```
<?xml version="1.0" encoding="UTF-8"?>
<CompanySettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <fiscalYear>
        <fiscalYearNameBasedOn>endingMonth</fiscalYearNameBasedOn>
        <startMonth>January</startMonth>
    </fiscalYear>
</CompanySettings>
```

SEE ALSO:

Settings

ContractSettings

Represents contract settings. For more information, see "Customizing Contract Settings" in the Salesforce online help.

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

There is one contract settings file stored in a file named `Contract.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ContractSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
autoCalculateEndDate	boolean	Indicates whether the end date of a contract is automatically calculated (true) or not (false).
notifyOwnersOnContractExpiration	boolean	Indicates whether account and contract owners are automatically sent email notifications when a contract expires (true) or not (false).

Declarative Metadata Sample Definition

This is a sample contract settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContractSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <autoCalculateEndDate>true</autoCalculateEndDate>
    <notifyOwnersOnContractExpiration>false</notifyOwnersOnContractExpiration>
</ContractSettings>
```

SEE ALSO:

Settings

EntitlementSettings

Represents an organization's entitlement settings.

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

EntitlementSettings values are stored in the Entitlements.settings file in the settings directory. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

EntitlementSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
assetLookupLimitedToActiveEntitlementsOnAccount	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the active entitlements on the case's account (true) or not (false).
assetLookupLimitedToActiveEntitlementsOnContact	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the active entitlements on the case's contact (true) or not (false).

Field Name	Field Type	Description
assetLookupLimitedToSameAccount	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the case's account (true) or not (false).
assetLookupLimitedToSameContact	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the case's contact (true) or not (false).
enableEntitlements	boolean	Indicates whether entitlements are enabled (true) or not (false).
enableEntitlementVersioning	boolean	Indicates whether entitlement versioning is enabled (true) or not (false). This field is available in API version 28.0 and later.
entitlementLookupLimitedToActiveStatus	boolean	Indicates whether entitlements-related lookup filters on cases return only active entitlements (true) or not (false).
entitlementLookupLimitedToSameAccount	boolean	Indicates whether entitlements-related lookup filters on cases return only the entitlements related to the case's account (true) or not (false).
entitlementLookupLimitedToSameAsset	boolean	Indicates whether entitlements-related lookup filters on cases return only the entitlements related to the case's asset (true) or not (false).
entitlementLookupLimitedToSameContact	boolean	Indicates whether entitlements-related lookup filters on cases return only the entitlements related to the case's contact (true) or not (false).

Declarative Metadata Sample Definition

This is a sample entitlements settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EntitlementSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <assetLookupLimitedToActiveEntitlementsOnAccount>
        false
    </assetLookupLimitedToActiveEntitlementsOnAccount>
    <assetLookupLimitedToActiveEntitlementsOnContact>
        false
    </assetLookupLimitedToActiveEntitlementsOnContact>
    <assetLookupLimitedToSameAccount>
        false
    </assetLookupLimitedToSameAccount>
    <assetLookupLimitedToSameContact>
        false
    </assetLookupLimitedToSameContact>
    <enableEntitlements>
        true
```

```
    </enableEntitlements>
    <entitlementLookupLimitedToActiveStatus>
        false
    </entitlementLookupLimitedToActiveStatus>
    <entitlementLookupLimitedToSameAccount>
        false
    </entitlementLookupLimitedToSameAccount>
    <entitlementLookupLimitedToSameAsset>
        false
    </entitlementLookupLimitedToSameAsset>
    <entitlementLookupLimitedToSameContact>
        false
    </entitlementLookupLimitedToSameContact>
</EntitlementSettings>
```

SEE ALSO:

Settings

ForecastingSettings

Represents the Collaborative Forecasts settings options. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: This information only applies to Collaborative Forecasts.

File Suffix and Directory Location

ForecastingSettings values are stored in a single file named `Forecasting.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ForecastingSettings components are available in API version 28 and later. The structure of the ForecastingSettings type changed significantly in API version 30.0.

Fields

Field Name	Field Type	Description
`displayCurrency`	DisplayCurrency (enumeration of type string)	The currency for displaying forecasts; either the organization's corporate currency or each forecast owner's personal currency setting. This is the default currency used in Collaborative Forecasts and selected in setup. The selection must be one of the currencies enabled for use in the organization, and only one selection is allowed. The default is `Corporate`. The valid values are: • `Corporate` • `Personal`

Field Name	Field Type	Description
`enableForecasts`	boolean	Indicates if Collaborative Forecasts is enabled or not. Set to `true` to enable Collaborative Forecasts and `false` to disable the functionality. Warning: Disabling Forecasts can result in data loss. Refer to the online Help before disabling any functionality.
`forecastingTypeSettings`	ForecastingTypeSettings[]	A list of forecast types. For field values, see ForecastingTypeSettings. The maximum number of forecast types is four.
`forecastingCategoryMappings`	ForecastingCategoryMappings[]	A list of mappings associating forecast types with forecast rollups.

ForecastingTypeSettings

The settings for each forecast type. An organization can have up to 4 forecast types active. Omitting a previously enabled forecast type that has a minimum API version less than or equal to the metadata package version deletes its quota and adjustment data from the organization.

Warning: Omitting a forecast type field from the XML can deactivate that forecast type: if the forecast type was available in the release specified by the XML package version, that forecast type is deactivated and its quota and adjustment data are deleted.

Field Name	Field Type	Description
`active`	boolean	This indicates whether the forecast type specified in the `name` field is active. Note: Setting the `active` field to false purges all forecasting data, adjustments, and quotas for the forecast type. When `active` is set to true, some values on the Forecasts tab may not appear immediately. An in-process icon appears to indicate that the values are being calculated.
`adjustmentsSettings`	AdjustmentsSettings	This enables or disables the Forecasts adjustments option in Forecasts.
`forecastRangeSettings`	ForecastRangeSettings	The default periods and range selections in Collaborative Forecasts.
`name`	string	The name of the forecast type. Each forecast type requires a specific string. Valid values include: • `OpportunityRevenue` : Opportunities - Revenue • `OpportunityQuantity` : Opportunities - Quantity • `OpportunitySplitRevenue` : Opportunity Revenue Splits - Revenue • `OpportunityOverlayRevenue` : Opportunity Overlay Splits - Revenue • `OpportunityLineItemRevenue` : Product Families - Revenue • `OpportunityLineItemQuantity` : Product Families - Quantity

Field Name	Field Type	Description
		• The name of a custom opportunity split type that has been enabled as a forecast type. Custom split types are based on currency fields, which can only contain revenue amounts.
`opportunityListFields SelectedSettings`	OpportunityListFields SelectedSettings	The fields selected to appear in the opportunity pane of the forecast page for the forecast type. `Opportunity Name` is required. You can select up to 15 fields.
`quotasSettings`	QuotasSettings	This enables or disables the quota option in Forecasts.
`forecastedCategoryApiNames`	string	This field appears four times to specify the four forecast rollup categories used in the organization, for either cumulative forecast rollups, or individual forecast category rollups. Valid values for organizations using cumulative forecast rollups: • openpipeline • bestcaseforecast • commitforecast • closedonly Valid values for organizations using individual forecast category rollups: • pipelineonly • bestcaseonly • commitonly • closedonly Changing from one set of four values to the other changes the organization setting for Enable Cumulative Forecast Rollups in Setup. If this field is omitted, the setting is not changed.
`displayedCategoryApiNames`	string	This read-only field appears four times to specify the four forecast rollup categories displayed in the Forecasts tab, for either cumulative forecast rollups, or individual forecast category rollups. Always use the same 4 values for both `displayedCategoryApiNames` and `forecastedCategoryApiNames`. Valid values for organizations using cumulative forecast rollups: • openpipeline • bestcaseforecast • commitforecast • closedonly Valid values for organizations using individual forecast category rollups: • pipelineonly • bestcaseonly • commitonly • closedonly

Field Name	Field Type	Description
managerAdjustableCategoryApiNames	string	This read-only field appears twice to specify the two forecast rollup categories that forecast managers can adjust in the organization for either cumulative forecast rollups or individual forecast category rollups. This field can only be used when the enableAdjustments field contains a value of true. If both the managerAdjustableCategoryApiNames and ownerAdjustableCategoryApiNames fields are being used, they must contain the same two values. Their values must also be consistent with the values of the enableAdjustments and enableOwnerAdjustments fields. Valid values for organizations using cumulative forecast rollups: • bestcaseforecast • commitforecast Valid values for organizations using individual forecast category rollups: • bestcaseonly • commitonly
ownerAdjustableCategoryApiNames	string	This read-only field appears twice to specify the two forecast rollup categories that forecast owners can adjust in the organization, for either cumulative forecast rollups, or individual forecast category rollups. This field can only be used when the enableOwnerAdjustments field contains a value of true. If both the managerAdjustableCategoryApiNames and ownerAdjustableCategoryApiNames fields are being used, they must contain the same two values. Their values must also be consistent with the values of the enableAdjustments and enableOwnerAdjustments fields. Valid values for organizations using cumulative forecast rollups: • bestcaseforecast • commitforecast Valid values for organizations using individual forecast category rollups: • bestcaseonly • commitonly

AdjustmentsSettings

The adjustment options for Collaborative Forecasts.

Field	Field Type	Description
`enableAdjustments`	boolean	Set to `true` to enable Collaborative Forecasts manager adjustments and `false` to disable them. All forecast types must contain the same `enableAdjustments` value. Warning: Disabling adjustments results in Collaborative Forecasts adjustment data being purged.
`enableOwnerAdjustments`	boolean	Set to `true` to enable Collaborative Forecasts owner adjustments and `false` to disable them. All forecast types must contain the same `enableAdjustments` value. Warning: Disabling adjustments results in Collaborative Forecasts adjustment data being purged.

ForecastRangeSettings

The default periods and range selections in Collaborative Forecasts. Users can forecast up to 12 months or eight quarters in the future or past. If your forecast range includes the current month or quarter, the Forecasts page displays the current month or quarter by default. If not, then the first month or quarter of the range is selected by default. All forecast types must contain the same `forecastRangeSettings` field values.

Warning: If you change the period setting from monthly to quarterly or quarterly to monthly, or you change the standard fiscal year, all adjustments and quotas are purged. If you enable custom fiscal years, creating the first custom fiscal year deletes any quotas and adjustments in the corresponding and subsequent standard fiscal years. These changes trigger a forecast recalculation that can take significant time, depending on the quantity of your data.

Field	Field Type	Description
`beginning`	int	Indicates the beginning month or quarter to display by default.
`displaying`	int	Indicates the number of months or quarters to display by default. The maximum number of months is 12 and quarters is 8.
`periodType`	PeriodTypes (enumeration of type string)	Indicates what type of period to use. Valid values are: • `Month` • `Quarter`

OpportunityListFieldsSelectedSettings

The fields selected to appear in the opportunity pane of the forecast page for the forecast type. `Opportunity Name` is required. You can select up to 15 fields.

Field	Field Type	Description
`field`	string	Specifies a field name to display in the opportunity pane.

QuotasSettings

QuotasSettings indicates if quotas is available in Collaborative Forecasts.

Field	Field Type	Description
`showQuotas`	boolean	Set to `true` to enable quotas. All forecast types must contain the same `showQuotas` field value.

ForecastingCategoryMappings

The forecasting category mappings for Collaborative Forecasts. This subtype appears eight times within the `ForecastingSettings` type. Each occurrence includes fields that specify a type of forecast category rollup, which forecast categories each rollup includes, and the weight of each forecast category in the rollup. Organizations using either cumulative forecast rollups or individual forecast category columns must include all eight occurrences of this subtype.

Field	Field Type	Description
`forecastingItemCategoryApiName`	string	This field specifies the API name of the rollup type. The valid values are: • openpipeline • bestcaseforecast • commitforecast • pipelineonly • bestcaseonly • commitonly • closedonly • omittedonly
`weightedSourceCategories`	WeightedSourceCategories[]	This field can occur more than once when specifying more than one forecast category to include in the rollup type. Each occurrence contains two subfields that specify a forecast category to include in the forecast rollup type and its weight. Some rollup types include more than one forecast category. This list shows the forecast categories that are included in each rollup type. • Rollup: openpipeline, Forecast categories: pipeline, best case, commit • Rollup: bestcaseforecast, Forecast categories: best case, commit, closed • Rollup: commitforecast, Forecast categories: commit, closed • Rollup: pipelineonly, Forecast categories: pipeline • Rollup: bestcaseonly, Forecast categories: best case • Rollup: commitonly, Forecast categories: commit • Rollup: closedonly, Forecast categories: closed • Rollup: omittedonly, Forecast categories: omitted

WeightedSourceCategories

This field can occur more than once when specifying more than one forecast category to include in the rollup type. Each occurrence contains two subfields that specify a forecast category to include in the forecast rollup type and its weight. Some rollup types include more than one forecast category. This table shows the forecast categories that are included in each rollup type.

Field	Field Type	Description
`sourceCategoryApiName`	string	Specifies the API name of a forecast category to include in the rollup type. The valid values are. • pipeline • best case • commit • closed • omitted
`weight`	double	Specifies the weight given to the forecast category when calculating the forecast for the rollup type. The only supported value is 1.0.

Declarative Metadata Sample Definition

The following is an example of a ForecastingSettings component that enables the Opportunity-Revenue and Product Family-Quantity forecast types:

```
<?xml version="1.0" encoding="UTF-8"?>
<ForecastingSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableForecasts>true</enableForecasts>
    <forecastingTypeSettings>
        <active>true</active>
        <adjustmentsSettings>
            <enableAdjustments>true</enableAdjustments>
        </adjustmentsSettings>
        <name>OpportunityRevenue</name>
        <forecastRangeSettings>
            <beginning>0</beginning>
            <displaying>6</displaying>
            <periodType>Month</periodType>
        </forecastRangeSettings>
        <opportunityListFieldsSelectedSettings>
            <field>OPPORTUNITY.NAME</field>
        </opportunityListFieldsSelectedSettings>
        <quotasSettings>
            <showQuotas>true</showQuotas>
        </quotasSettings>
    </forecastingTypeSettings>
    <forecastingTypeSettings>
        <active>false</active>
        <adjustmentsSettings>
```

```
            <enableAdjustments>true</enableAdjustments>
        </adjustmentsSettings>
        <name>OpportunityLineItemQuantity</name>
        <forecastRangeSettings>
            <beginning>0</beginning>
            <displaying>6</displaying>
            <periodType>Month</periodType>
        </forecastRangeSettings>
        <opportunityListFieldsSelectedSettings>
            <field>OPPORTUNITY.NAME</field>
        </opportunityListFieldsSelectedSettings>
        <quotasSettings>
            <showQuotas>true</showQuotas>
        </quotasSettings>
        <displayedCategoryApiNames>pipelineonly</displayedCategoryApiNames>
        <displayedCategoryApiNames>bestcaseonly</displayedCategoryApiNames>
        <displayedCategoryApiNames>commitonly</displayedCategoryApiNames>
        <displayedCategoryApiNames>closedonly</displayedCategoryApiNames>
        <forecastedCategoryApiNames>commitonly</forecastedCategoryApiNames>
        <forecastedCategoryApiNames>closedonly</forecastedCategoryApiNames>
        <forecastedCategoryApiNames>bestcaseonly</forecastedCategoryApiNames>
        <forecastedCategoryApiNames>pipelineonly</forecastedCategoryApiNames>
        <managerAdjustableCategoryApiNames>commitonly</managerAdjustableCategoryApiNames>

       <managerAdjustableCategoryApiNames>bestcaseonly</managerAdjustableCategoryApiNames>

        <ownerAdjustableCategoryApiNames>commitonly</ownerAdjustableCategoryApiNames>
        <ownerAdjustableCategoryApiNames>bestcaseonly</ownerAdjustableCategoryApiNames>
    </forecastingTypeSettings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>commitonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>closedonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>closed</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>bestcaseforecast</forecastingItemCategoryApiName>

        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>best case</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
```

```
        <weightedSourceCategories>
            <sourceCategoryApiName>closed</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>omittedonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>omitted</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>openpipeline</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>best case</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>pipeline</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>bestcaseonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>best case</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>commitforecast</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>closed</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>pipelineonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>pipeline</sourceCategoryApiName>
            <weight>1.0</weight>
```

```
        </weightedSourceCategories>
    </forecastingCategoryMappings>
```

SEE ALSO:

Settings

IdeasSettings

Represents the metadata used to manage settings for Ideas.

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

IdeasSettings is stored in one file named `Ideas.settings` in the `settings` folder of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

IdeasSettings is available in API version 27.0 and later.

Ideas

Represents settings for Ideas and Idea Themes.

Fields

Field Name	Field Type	Description
`enableIdeaThemes`	boolean	Indicates whether Idea Themes is enabled (`true`) or not (`false`).
`enableIdeas`	boolean	Indicates whether Ideas is enabled (`true`) or not (`false`).
`enableIdeasReputation`	boolean	Indicates whether Reputation is enabled (`true`) or not (`false`). You can't enable IdeasReputation without enabling the Ideas Reputation permission in your organization. This field is available in API version 28.0 and later.
`enableChatterProfile`	boolean	Indicates that the Chatter user profile is used for Ideas user profiles. If `enableChatterProfile` is `true`, the `ideasProfilePage` value must not be specified. If `enableChatterProfile` is `false`, then specify a `ideasProfilePage` value, otherwise the Ideas zone profile is used. This field is available in API version 29.0 and later.
`ideasProfilePage`	string	The name of the Visualforce page to use for a custom Ideas user profile, if `enableChatterProfile` is `false`. If `enableChatterProfile` is `false`, then specify a `ideasProfilePage` value, otherwise the Ideas zone profile is used. This field is available in API version 29.0 and later.

Field Name	Field Type	Description
halfLife	double	Indicates how quickly old ideas drop in ranking on the Popular Ideas subtab. The half-life setting determines how the number of days after which old ideas drop in ranking on the Popular Ideas subtab, to make room for ideas with more recent votes. A shorter half-life moves older ideas down the page faster than a longer half-life.

Declarative Metadata Sample Definition

The following is an example `ideas.settings` metadata file:

```
<?xml version="1.0" encoding="UTF-8"?>
<IdeasSettings xmlns="http://soap.sforce.com/2006/04/metadata">
   <enableIdeaThemes>true</enableIdeaThemes>
   <enableIdeas>true</enableIdeas>
   <enableIdeasReputation>true</enableIdeasReputation>
   <enableChatterProfile>false</enableChatterProfile>
   <ideasProfilePage>name of Visualforce page</ideasProfilePage>
   <halfLife>2.6</halfLife>
</IdeasSettings>
```

SEE ALSO:

Settings

KnowledgeSettings

Represents the metadata used to manage settings for Salesforce Knowledge. This type 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

KnowledgeSettings values are stored in a single file named `Knowledge.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

KnowledgeSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
answers	KnowledgeAnswerSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Answers.

Field Name	Field Type	Description
`cases`	KnowledgeCaseSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Cases.
`defaultLanguage`	string	Required. The default language for Salesforce Knowledge. Use the abbreviation for the language, for example, en_US for United States English.
`languages`	KnowledgeLanguageSettings	A list of languages enabled for Salesforce Knowledge.
`enableChatterQuestionKBDeflection`	boolean	Indicates whether tracking for case deflection via Chatter is enabled (`true`) or not (`false`).
`enableCreateEditOnArticlesTab`	boolean	Indicates whether users can create and edit articles on the articles tab (`true`) or not (`false`).
`enableExternalMediaContent`	boolean	Indicates whether connecting to external media is enabled (`true`) or not (`false`).
`enableKnowledge`	boolean	Indicates whetherSalesforce Knowledge is enabled (`true`) or not (`false`).
`showArticleSummariesCustomerPortal`	boolean	Indicates whether article summaries appear in the Customer Portal (`true`) or not (`false`).
`showArticleSummariesInternalApp`	boolean	Indicates whether article summaries appear in the internal knowledge base (`true`) or not (`false`).
`showArticleSummariesPartnerPortal`	boolean	Indicates whether article summaries appear in the partner portal (`true`) or not (`false`).
`showValidationStatusField`	boolean	Indicates whether validation status appears on articles (`true`) or not (`false`).

KnowledgeAnswerSettings

Represents the metadata used to manage settings for Salesforce Knowledge and Answers.

Field Name	Field Type	Description
`assignTo`	string	Specifies the username an article is assigned to from Answers.
`defaultArticleType`	string	The default article type for articles created from Answers. Uses the API name of the article type.
`enableArticleCreation`	boolean	Indicates whether users can create articles from Answers (`true`) or not (`false`).

KnowledgeCaseSettings

Represents the metadata used to manage settings for Salesforce Knowledge and Cases.

Field Name	Field Type	Description
`articlePDFCreationProfile`	string	The profile used to create a PDF of an article from Cases.
`articlePublicSharingSites`	KnowledgeSitesSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Sites.
`articlePublicSharingCommunities`	KnowledgeSitesSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Communities.
`articlePublicSharingSitesChatterAnswers`	KnowledgeSitesSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Sites with Chatter Answers.
`assignTo`	string	Specifies the username an article is assigned to from Cases.
`customizationClass`	string	Specifies the Apex class used for customization.
`defaultContributionArticleType`	string	The default article type for articles created from Cases.
`editor`	KnowledgeCaseEditor (enumeration of type string)	Indicates the rich text editor type. Valid values are: • `simple` • `standard`
`enableArticleCreation`	boolean	Indicates whether users can create articles from Cases (`true`) or not (`false`). Controls whether other fields on KnowledgeCaseSettings can be set.
`enableArticlePublicSharingSites`	boolean	Indicates whether articles can be shared via a public site (URL) from Cases (`true`) or not (`false`).
`useProfileForPDFCreation`	boolean	Indicates whether a profile is used to create a PDF of an article from Cases (`true`) or not (`false`).

KnowledgeSitesSettings

Represents the metadata used to manage settings for Salesforce Knowledge and Sites.

Field Name	Field Type	Description
`site`	string[]	Specifies the site used for Salesforce Knowledge and Sites.

KnowledgeLanguageSettings

A list of languages enabled for Salesforce Knowledge. KnowledgeLanguageSettings is available in API version 28.0 and later.

Field Name	Field Type	Description
language	KnowledgeLanguage	Represents the metadata used to manage settings for the languages enabled for Salesforce Knowledge.

KnowledgeLanguage

Represents the metadata used to manage settings for the languages enabled for Salesforce Knowledge. KnowledgeLanguage is available in API version 28.0 and later.

Field Name	Field Type	Description
active	boolean	Indicates whether the language is enabled (true) or not (false).
defaultAssignee	string	The default assignee for articles in the language.
defaultAssigneeType	KnowledgeLanguageLookupValueType (enumeration of type string)	Indicates the default assignee type. Valid values are: • user • queue
defaultReviewer	string	The default reviewer for articles in the language.
defaultReviewerType	KnowledgeLanguageLookupValueType (enumeration of type string)	Indicates the default reviewer type. Valid values are: • user • queue
name	string	The code for the language name, for example: English is en. See "What languages does Salesforce support?" in the Salesforce online help for a list of supported languages and their codes.

Declarative Metadata Sample Definition

This is a sample Knowledge settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<KnowledgeSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <answers>
        <enableArticleCreation>false</enableArticleCreation>
    </answers>
    <cases>
        <articlePDFCreationProfile>partner portal knowledge
profile</articlePDFCreationProfile>
        <articlePublicSharingSites>
            <site>KnowledgeSite</site>
            <site>PKB2Site</site>
            <site>ChatterAnswersSite</site>
        </articlePublicSharingSites>
        <articlePublicSharingSitesChatterAnswers>
```

```
            <site>ChatterAnswersSite</site>
        </articlePublicSharingSitesChatterAnswers>
        <assignTo>testall@kb.org</assignTo>
        <defaultContributionArticleType>Support</defaultContributionArticleType>
        <editor>simple</editor>
        <enableArticleCreation>true</enableArticleCreation>
        <enableArticlePublicSharingSites>true</enableArticlePublicSharingSites>
        <useProfileForPDFCreation>true</useProfileForPDFCreation>
    </cases>
    <defaultLanguage>ja</defaultLanguage>
    <enableCreateEditOnArticlesTab>true</enableCreateEditOnArticlesTab>
    <enableExternalMediaContent>true</enableExternalMediaContent>
    <enableKnowledge>true</enableKnowledge>
    <showArticleSummariesCustomerPortal>true</showArticleSummariesCustomerPortal>
    <showArticleSummariesInternalApp>true</showArticleSummariesInternalApp>
    <showArticleSummariesPartnerPortal>true</showArticleSummariesPartnerPortal>
    <showValidationStatusField>true</showValidationStatusField>
</KnowledgeSettings>
```

SEE ALSO:

Settings

LiveAgentSettings

Represents an organization's Live Agent settings, such as whether or not Live Agent is enabled. This type 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 because 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
`enableLiveAgent`	boolean	Indicates whether Live Agent is enabled (`true`) or not (`false`).

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 "Manage Salesforce Classic Mobile Devices" and "Chatter Mobile for BlackBerry" in the Salesforce 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 because 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
`chatterMobile`	ChatterMobileSettings	The settings for Chatter mobile devices.
`dashboardMobile`	DashboardMobileSettings	The settings for dashboards on mobile devices.
`salesforceMobile`	SFDCMobileSettings	The settings for general users on mobile devices.
`touchMobile` (Deprecated)	TouchMobileSettings	The settings for touch on mobile devices.

ChatterMobileSettings

Represents your organization's Chatter Mobile settings.

Field	Field Type	Description
`IPadAuthorized`	boolean	Indicates whether iPad devices are enabled for Chatter Mobile (`true`) or not (`false`).

Field	Field Type	Description
`IPhoneAuthorized`	boolean	Indicates whether iPhone devices are enabled for Chatter Mobile (`true`) or not (`false`).
`androidAuthorized`	boolean	Indicates whether Android devices are enabled for Chatter Mobile (`true`) or not (`false`).
`blackBerryAuthorized`	boolean	Indicates whether Blackberry devices are enabled for Chatter Mobile (`true`) or not (`false`).
`enableChatterMobile`	boolean	Indicates whether Chatter Mobile has been enabled for your organization (`true`) or not (`false`). 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.
`enablePushNotifications`	boolean	Indicates whether Chatter push notifications have been enabled for your organization (`true`) or not (`false`)
`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: • `Never` • `OneMinute` • `FiveMinutes` • `TenMinutes` • `ThirtyMinutes`

DashboardMobileSettings

Represents your organization's Mobile Dashboards iPad app settings.

Field	Field Type	Description
`enableDashboardIPadApp`	boolean	Indicates whether Mobile Dashboards iPad app has been enabled for your organization (`true`) or not (`false`)

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 `true` only if you want to prevent your users from switching devices without administrative intervention..
`enableMobileLite`	boolean	Indicates whether your organization has Mobile Lite enabled (`true`) or not (`false`).

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 (`true`) or not (`false`).
`enableTouchAppIPad`	boolean	Indicates whether your organization has the Salesforce Touch downloadable app enabled (`true`) or not (`false`)

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>
        <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

NameSettings

Enables or disables middle name and suffix attributes for the following person objects: Contact, Lead, Person Account, and User.

File Suffix and Directory Location

NameSettings values are stored in a single file named `Name.settings` in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

NameSettings components are available in API version 31.0 and later.

Fields

Field Name	Field Type	Description
`enableMiddleName`	boolean	Indicates whether middle names are enabled (`true`) or disabled (`false`) for person objects.
`enableNameSuffix`	boolean	Indicates whether suffixes are enabled (`true`) or disabled (`false`) for person objects.

Declarative Metadata Sample Definition

The following is an example of a NameSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<NameSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableMiddleName>true</enableMiddleName>
    <enableNameSuffix>false</enableNameSuffix>
</NameSettings>
```

The following is an example `package.xml` manifest that references the NameSettings definitions.

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

```
    <version>31.0</version>
</Package>
```

OpportunitySettings

Represents organization preferences for features such as automatic opportunity updates and similar-opportunity filters. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Opportunities values are stored in a single file named `Opportunity.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OpportunitySettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`enableUpdateReminders`	boolean	Lets users enable automatic, scheduled updates on opportunities.
`autoActivateNewReminders`	boolean	Automatically uses scheduled updates for new opportunities.
`enableFindSimilarOpportunities`	boolean	Lets users see related or similar existing opportunities.
`findSimilarOppFilter`	multipicklist	Defines parameters for similar opportunities.
`enableOpportunityTeam`	boolean	Lets users associate team members with opportunities.
`promptToAddProducts`	boolean	Prompts users to add related products to an opportunity.

FindSimilarOppFilter

Defines whether to match by entire columns or fields.

Field	Field Type	Description
`similarOpportunitiesDisplayColumns`	string	The columns to compare.
`similarOpportunitiesMatchFields`	string	The fields to compare.

Declarative Metadata Sample Definition

The following is an example of the package file.

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

```
    <types>
        <members>Opportunity</members>
        <name>Settings</name>
    </types>
    <version>28.0</version>
</Package>
```

The package file references the following Opportunity.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<OpportunitySettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <autoActivateNewReminders>true</autoActivateNewReminders>
    <enableFindSimilarOpportunities>true</enableFindSimilarOpportunities>
    <findSimilarOppFilter>

<similarOpportunitiesMatchFields>OPPORTUNITY.Account</similarOpportunitiesMatchFields>

<similarOpportunitiesMatchFields>OPPORTUNITY.OpportunityCompetitors</similarOpportunitiesMatchFields>

        <similarOpportunitiesMatchFields>CustomField__c</similarOpportunitiesMatchFields>


<similarOpportunitiesDisplayColumns>CustomField__c</similarOpportunitiesDisplayColumns>
    </findSimilarOppFilter>
    <enableOpportunityTeam>true</enableOpportunityTeam>
    <enableUpdateReminders>true</enableUpdateReminders>
    <promptToAddProducts>false</promptToAddProducts>
</OpportunitySettings>
```

OrderSettings

Represents order settings. This type extends the Metadata metadata type and inherits its `fullName` field. For more information, see "Customizing Order Settings" in the Salesforce Help.

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

There is one OrderSettings component in a file named `Order.settings` in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OrderSettings components are available in API version 30.0 and later.

Fields

Field Name	Field Type	Description
`enableNegativeQuantity`	boolean	Indicates whether users in the organization can add order products with quantities of less than zero (`true`) or not (`false`). To enable this preference, `enableOrders` must be set to `true`.
`enableOrders`	boolean	Indicates whether orders are enabled for the organization (`true`) or not (`false`).
`enableReductionOrders`	boolean	Indicates whether reduction orders are enabled for the organization (`true`) or not (`false`). For more information, see "Reduction Orders Overview" in the Salesforce Help. To enable this preference, `enableOrders` must be set to `true`.

Declarative Metadata Sample Definition

This is a sample OrderSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<OrderSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableOrders>true</enableOrders>
    <enableReductionOrders>false</enableReductionOrders>
    <enableNegativeQuantity>true</enableNegativeQuantity>
</OrderSettings>
```

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>Order</members>
        <name>Settings</name>
    </types>
    <version>30.0</version>
</Package>
```

PathAssistantSettings

Represents the Sales Path preference setting.This type 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

PathAssistantSettings components have the suffix `.settings` and are stored in the `settings` folder.

Version

PathAssistantSettings components are available in API version 34.0 and later.

Fields

Field Name	Field Type	Description
pathAssistantForOpportunityEnabled	boolean	Determines whether the preference is enabled for Sales Path in Opportunity or not.

Declarative Metadata Sample Definition

The following is an example of a PathAssistantSettings component.

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

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>PathAssistant</members>
        <name>Settings</name>
    </types>
    <version>API</version>
</Package
```

ProductSettings

Represents organization preferences for quantity schedules, revenue schedules, and active flag interaction with prices. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ProductSettings values are stored in a single file named `Product.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ProductSettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
enableCascadeActivateToRelatedPrices	boolean	When changing active flag on a product record, automatically updates active flag on related prices.
enableQuantitySchedule	boolean	Enables quantity schedules for products.
enableRevenueSchedule	boolean	Enables revenue schedules for products.

Declarative Metadata Sample Definition

The following is an example of the package file.

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

The package file references the following Product.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ProductSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableCascadeActivateToRelatedPrices>true</enableCascadeActivateToRelatedPrices>
    <enableQuantitySchedule>false</enableQuantitySchedule>
    <enableRevenueSchedule>false</enableRevenueSchedule>
</ProductSettings>
```

QuoteSettings

Enables or disables Quotes, which show proposed prices for products and services. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

QuoteSettings values are stored in a single file named `Quote.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

QuoteSettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
enableQuote	boolean	When set to true, users can access Quotes.

Declarative Metadata Sample Definition

The following is an example of the package file.

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

The package file references the following Quote.settings file.

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

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 because 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
networkAccess	NetworkAccess	The trusted IP address ranges from which users can always log in without requiring computer activation.

Field Name	Field Type	Description
passwordPolicies	PasswordPolicies	The requirements for passwords and logins, and assistance with retrieving forgotten passwords.
sessionSettings	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[]	The trusted IP address ranges from which users can always log in without requiring computer activation. Note: In order to add an IP range, 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>).

IpRange

Defines a range of trusted IP addresses for network access.

Field	Field Type	Description
description	string	The description of the trusted IP range. Use this field to identify the range, such as which corporate network corresponds to this range. This field is available in API version 34.0 and later.
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.

Field	Field Type	Description
`complexity`	Complexity (enumeration of type string)	Required. The requirement for which types of characters must be used in a user's password. Valid values are: • `NoRestriction`—allows any password value and is the least secure option. • `AlphaNumeric`—requires at least one alphabetic character and one number. This value is the default value. • `SpecialCharacters`—requires at least one alphabetic character, one number, and one of the following characters: `! # $ % - _ = + < >`. • `UpperLowerCaseNumeric`—requires at least one number, one uppercase letter, and one lowercase letter. This value is available in API version 31.0 and later. • `UpperLowerCaseNumericSpecialCharacters`—requires at least one number, one uppercase letter, and one lowercase letter, and one of the following characters: `! # $ % - _ = + < >`. This value is available in API version 31.0 and later.
`expiration`	Expiration (enumeration of type string)	Required. The length of time until user passwords expire and must be changed. Valid values are: • `Never` • `ThirtyDays` • `SixtyDays` • `NinetyDays`. This value is the default value. • `SixMonths` • `OneYear`
`minimumPasswordLifetime`	boolean	Indicates whether a one-day minimum password lifetime is required (`true`) or not (`false`). This field is available in API version 31.0 and later.
`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 `24` passwords remembered. The maximum value of 24 applies to API version 31.0 and later. In earlier versions, the maximum value is 16. The default value is `3`.
`lockoutInterval`	LockoutInterval (enumeration of type string)	Required. The duration of the login lockout. Valid values are: • `FifteenMinutes`. This value is the default value. • `ThirtyMinutes`

Field	Field Type	Description
		• `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: • `NoLimit` • `ThreeAttempts` • `FiveAttempts` • `TenAttempts`. This value is the default value.
`minimumPasswordLength`	string	Required. The minimum number of characters required for a password. Valid values are from 5 to 50. The default value is 8. This field is available in API version 35.0 and later. Before API version 35.0, specify minimum password length with the enumeration `minPasswordLength`, with valid values `FiveCharacters`, `EightCharacters` (default), `TenCharacters`, `TwelveCharacters` (API version 31.0 and later), and `FifteenCharacters` (API version 34.0 and later).
`obscureSecretAnswer`	boolean	Hides the secret answer associated with a password (`true`) or not (`false`). 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.
`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: • `None` • `DoesNotContainPassword`. This value 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 (true) or enabled (false).
enableCSPOnEmail	boolean	Indicates whether a content security policy is enabled for the email template. A content security policy helps prevent cross-site scripting attacks by whitelisting sources of images and other content.
enableCSRFOnGet	boolean	Indicates whether Cross-Site Request Forgery (CSRF) protection on GET requests on non-setup pages is enabled (true) or disabled (false).
enableCSRFOnPost	boolean	Indicates whether Cross-Site Request Forgery (CSRF) protection on POST requests on non-setup pages is enabled (true) or disabled (false).
enableCacheAndAutocomplete	boolean	Indicates whether the user's browser is allowed to store user names and auto-fill the User Name field on the login page (true) or not (false).
enableClickjackNonsetupSFDC	boolean	Indicates whether clickjack protection for non-setup Salesforce pages is enabled (true) or disabled (false).
enableClickjackNonsetupUser	boolean	Indicates whether clickjack protection for customer Visualforce pages with standard headers turned on is enabled (true) or disabled (false).
enableClickjackNonsetupUserHeaderless	boolean	Indicates whether clickjack protection for customer Visualforce pages with standard headers turned off is enabled (true) or disabled (false). Available in API version 34.0 and later.
enableClickjackSetup	boolean	Indicates whether clickjack protection for setup pages is enabled (true) or disabled (false).
enablePostForSessions	boolean	Indicates whether cross-domain session information is exchanged using a POST request instead of a GET request, such as when a user is using a Visualforce page. In this context, POST requests are more secure than GET requests. Available in API version 31.0 and later.
enableSMSIdentity	boolean	Indicates whether users can receive a one-time PIN delivered via SMS (true) or not (false).
enforceIpRangesEveryRequest	boolean	If true, the IP addresses in Login IP Ranges are enforced when a user accesses Salesforce (on every page request), including access from a client application. If false, the IP addresses in Login IP Ranges are enforced only when a user logs in. This field affects all user profiles that have login IP restrictions. Available in API version 34.0 and later.
forceLogoutOnSessionTimeout	boolean	Indicates that when sessions time out for inactive users, current sessions become invalid. The browser refreshes and returns to

Field	Field Type	Description
		the login page. To access the organization, the user must log in again. Enabled (`true`) or not (`false`). Available in API version 31.0 and later.
`forceRelogin`	boolean	If `true`, 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. If `false`, the administrator is not required to log in again.
`lockSessionsToDomain`	boolean	Indicates whether the current UI session for a user, such as a community user, is associated with a specific domain. This check helps prevent unauthorized use of the session ID in another domain. The value is `true` by default for organizations created with the Spring '15 release or later. Available in API version 33.0 and later.
`lockSessionsToIp`	boolean	Indicates whether user sessions are locked to the IP address from which the user logged in (`true`) or not (`false`).
`logoutURL`	string	The URL to which users are redirected when they log out of Salesforce. If no value is specified, the default is `https://login.salesforce.com` unless MyDomain is enabled. If My Domain is enabled, the default is `https://`*`customdomain`*`.my.salesforce.com`. Available in API version 34.0 and later.
`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: • `FifteenMinutes` • `ThirtyMinutes` • `SixtyMinutes` • `TwoHours` • `FourHours` • `EightHours` • `TwelveHours`

Declarative Metadata Sample Definition

The following 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>
        <minimumPasswordLength>10</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

Territory2Settings

Represents the metadata for the default settings for Territory Management 2.0 users to access and modify records associated with sales territories. The standard record access settings apply to accounts and opportunities. If your organization uses `Private` default internal access for contacts or cases, you can also set access for those records. This type extends the Metadata metadata type and inherits its `fullName` field. Available only if Territory Management 2.0 has been enabled for your organization.

File Suffix and Directory Location

Territory2Settings components have the suffix `settings` and are stored in the `Settings` folder.

Version

Territory2Settings components are available in API version 32.0 and later.

Special Access Rules

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only territories that belong to the model in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.

Fields

Field Name	Field Type	Description
`defaultAccountAccessLevel`	string	The default level of access users will have to account records in territories: *`view`* and *`edit`* accounts assigned to territories or *`view`*, *`edit`*, *`transfer`*, and *`delete`* accounts assigned to territories.
`defaultCaseAccessLevel`	string	The default level of access users will have to case records in territories: *`view`* and *`edit`* accounts assigned to territories or *`view`*, *`edit`*, *`transfer`*, and *`delete`* accounts assigned to territories.
`defaultContactAccessLevel`	string	The default level of access users will have to contact records in territories: *`view`* and *`edit`* accounts assigned to territories or *`view`*, *`edit`*, *`transfer`*, and *`delete`* accounts assigned to territories.
`defaultOpportunityAccessLevel`	string	The default level of access users will have to opportunity records in territories: *`view`* and *`edit`* accounts assigned to territories or *`view`*, *`edit`*, *`transfer`*, and *`delete`* accounts assigned to territories.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2Settings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2Settings xmlns="http://soap.sforce.com/2006/04/metadata">
    <defaultAccountAccessLevel>Owner</defaultAccountAccessLevel>
    <defaultOpportunityAccessLevel>Read</defaultOpportunityAccessLevel>
    <defaultCaseAccessLevel>None</defaultCaseAccessLevel>
    <defaultContactAccessLevel>Edit</defaultContactAccessLevel>
</Territory2Settings>
```

Usage

Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.

SharedTo

SharedTo defines the sharing access for a list view or a folder. It can be used to specify the target and source for owner-based sharing rules. See "Sharing Considerations" and "What Is a Group?" in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

SharedTo is used with ListView, Folder, and SharingRules.

Version

SharedTo is available in API version 17.0 and later.

Fields

Field	Field Type	Description
`allCustomerPortalUsers`	string	A group containing all customer portal users. This field is available in API version 24.0 and later.
`allInternalUsers`	string	A group containing all internal and nonportal users. This field is available in API version 24.0 and later.
`allPartnerUsers`	string	A group containing all partner users. This field is available in API version 24.0 and later.
`group`	string[]	A list of groups with sharing access. Use this field instead of the `groups` field. This field is available in API version 22.0 and later.
`groups`	string[]	A list of groups with sharing access. Use the `group` field instead for API version 22.0 and later.
`managerSubordinates`	string[]	A list of users whose direct and indirect subordinates receive sharing access. This field is available in API version 24.0 and later.
`managers`	string[]	A list of users whose direct and indirect managers receive sharing access. This field is available in API version 24.0 and later.
`portalRole`	string[]	A list of groups with sharing access containing all users in a portal role. This field is available in API version 24.0 and later.
`portalRoleandSubordinates`	string[]	A list of groups with sharing access containing all users in a portal role or those under that role. This field is available in API version 24.0 and later.

Field	Field Type	Description
`role`	string[]	A list of roles with sharing access. Use this field instead of the `roles` field. This field is available in API version 22.0 and later.
`roleAndSubordinates`	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. If portal accounts are enabled, then all roles and portal accounts below each of these roles in the role hierarchy also have sharing access. Use this field instead of the `rolesAndSubordinates` field . This field is available in API version 22.0 and later.
`roleAndSubordinatesInternal`	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.
`roles`	string[]	A list of roles with sharing access. Use the `role` field instead for API version 22.0 and later.
`rolesAndSubordinates`	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. If portal accounts are enabled, then all roles and portal accounts below each of these roles in the role hierarchy also have sharing access. Use the `roleAndSubordinates` field instead for API version 22.0 and later.
`territories`	string[]	A list of territories with sharing access. Use the `territory` field instead for API version 22.0 and later.
`territoriesAndSubordinates`	string[]	A list of territories with sharing access. All territories below each of these territories in the territory hierarchy also have sharing access. Use the `territoryAndSubordinates` field instead for API version 22.0 and later.
`territory`	string[]	A list of territories with sharing access. Use this field instead of the `territories` field. This field is available in API version 22.0 and later.
`territoryAndSubordinates`	string[]	A list of territories with sharing access. All territories below each of these territories in the territory hierarchy also have sharing access. Use this field instead of the `territoriesAndSubordinates` field. This field is available in API version 22.0 and later.

Field	Field Type	Description
queue	string[]	A list of queues with sharing access. Applies only to lead, case, and CustomObject sharing rules. This field is available in API version 24.0 and later.

SharingBaseRule

Represents sharing rule settings such as access level and to whom access is granted. This type extends the Metadata metadata type and inherits its fullName field.

Note: You can't create a SharingBaseRule component directly. Use the components under SharingRules instead.

Version

SharingBaseRule replaces BaseSharingRule and is available in API version 33.0 and later.

Fields

Field	Field Type	Description
accessLevel	string	Required. The access level that the sharing rule grants.
accountSettings	AccountSharingRuleSettings[]	The access level for the account's children (case, contact, and opportunity).
description	string	Describes the sharing rule. Maximum of 1000 characters.
label	string	Required. Label for the sharing rule.
sharedTo	SharedTo	Required. Specifies who the record should be shared with.

AccountSharingRuleSettings

Defines the access level for the case, contact, and opportunity associated with the account.

Field	Field Type	Description
caseAccessLevel	string	Required. The access level that the user or group has to cases associated with the account. Possible values are: • None • Read

Field	Field Type	Description
		• Edit
`contactAccessLevel`	string	Required. The access level that the user or group has to contacts associated with the account. Possible values are: • None • Read • Edit
`opportunityAccessLevel`	string	Required. The access level that the user or group has to opportunities associated with the account. Possible values are: • None • Read • Edit

SharingRules

Represents the base container for sharing rules, which can be criteria-based, ownership-based, or territory-based. SharingRules enables you to share records with a set of users, using rules that specify the access level for the target user group.

This type extends the Metadata metadata type and inherits its `fullName` field. For more information, see "Sharing Rules" in the Salesforce online help.

In API version 33.0 and later, retrieving, deleting, or deploying of all sharing rules in an organization is available . Wildcard support is also available. You can't retrieve, delete, or deploy manual sharing rules or sharing rules by their type (owner, criteria-based, or territory).

Declarative Metadata File Suffix and Directory Location

In API version 33.0 and later, components are stored in the sharingRules folder and their file name matches the object name with the suffix `.sharingRules`. Criteria-based, owner-based, and territory-based sharing rules are all contained in a `object.sharingRule` file.

Prior to API version 33.0, SharingRules components are stored in their corresponding object directory and the file name matches the object name. For example, the `accountSharingRules` directory contains an `Account.sharingRules` file for account sharing rules. 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, but these components are no longer available in API version 33.0 and later: AccountSharingRules, CampaignSharingRules, CaseSharingRules, ContactSharingRules, LeadSharingRules, OpportunitySharingRules, AccountTerritorySharingRules, CustomObjectSharingRules, UserSharingRules.

In API version 33.0 and later, use SharingCriteriaRule, SharingOwnerRule and SharingTerritoryRule.

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 "Sharing Settings" in the Salesforce online help.

Field	Field Type	Description
`sharingCriteriaRules`	SharingCriteriaRule[]	An array of criteria-based sharing rules. Available in API version 33.0 and later.
`sharingOwnerRules`	SharingOwnerRule[]	An array of ownership-based sharing rules. Available in API version 33.0 and later.
`sharingTerritoryRules`	SharingTerritoryRule[]	An array of territory-based sharing rules. Available in API version 33.0 and later.

SharingCriteriaRule

Defines a criteria-based sharing rule. It extends SharingBaseRule and inherits all its fields. Available in API version 33.0 and later.

Field	Field Type	Description
`booleanFilter`	string	Advanced filter conditions that are specified for the sharing rule.
`criteriaItems`	FilterItem[]	An array of the boolean criteria (conditions) for the sharing rule.

SharingOwnerRule

Defines a ownership-based sharing rule. It extends SharingBaseRule and inherits all its fields. Available in API version 33.0 and later.

Field	Field Type	Description
`sharedFrom`	SharedTo	Required. Specifies the record owners.

SharingTerritoryRule

Defines a territory-based sharing rule. It extends SharingOwnerRule and inherits all its fields. Available in API version 33.0 and later.

AccountSharingRules

Represents the sharing rules for accounts. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	AccountCriteriaBasedSharingRule[]	List that defines user criteria-based rules.

Field	Field Type	Description
ownerRules	AccountOwnerSharingRule[]	List that defines user membership-based rules.

CampaignSharingRules

Represents the sharing rules for campaigns. It extends the SharingRules metadata type and inherits its fullName field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
criteriaBasedRules	CampaignCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
ownerRules	CampaignOwnerSharingRule[]	List that defines user membership-based rules.

CaseSharingRules

Represents the sharing rules for cases. It extends the SharingRules metadata type and inherits its fullName field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
criteriaBasedRules	CaseCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
ownerRules	CaseOwnerSharingRule[]	List that defines user membership-based rules.

ContactSharingRules

Represents the sharing rules for contacts. It extends the SharingRules metadata type and inherits its fullName field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
criteriaBasedRules	ContactCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
ownerRules	ContactOwnerSharingRule[]	List that defines user membership-based rules.

LeadSharingRules

Represents the sharing rules for leads. It extends the SharingRules metadata type and inherits its fullName field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
criteriaBasedRules	LeadCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
ownerRules	LeadOwnerSharingRule[]	List that defines user membership-based rules.

OpportunitySharingRules

Represents the sharing rules for opportunities. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	OpportunityCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	OpportunityOwnerSharingRule[]	List that defines user membership-based rules.

AccountTerritorySharingRules

Represents the sharing rules for account territories. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`rules`	AccountTerritorySharingRule[]	List that defines user membership-based rules. The list of acceptable values for the `sharedFrom` fields are: • `territory` • `territoryAndSubordinates`

CustomObjectSharingRules

Represents the sharing rules for custom objects. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	CustomObjectCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	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. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	UserCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`membershipRules`	UserMembershipSharingRule[]	List that defines user membership-based rules.

Declarative Metadata Sample Definition

For retrieving sharing rules, see `package.xml` sample at Sharing Rules.

The following sample XML definition represents a criteria-based sharing rule in API version 33.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <sharingCriteriaRules>
        <fullName>AccountCriteriaShareWithCEO</fullName>
        <accessLevel>Edit</accessLevel>
        <accountSettings>
            <caseAccessLevel>Read</caseAccessLevel>
            <contactAccessLevel>Edit</contactAccessLevel>
            <opportunityAccessLevel>Edit</opportunityAccessLevel>
        </accountSettings>
        <criteriaItems>
            <field>Name</field>
            <operation>startsWith</operation>
            <value>Test</value>
        </criteriaItems>
        <description>my account criteria rule description</description>
        <label>AccountCriteriaShareWithCEO</label>
        <sharedTo>
            <role>CEO</role>
        </sharedTo>
    </sharingCriteriaRules>
</SharingRules>
```

The following sample XML definition represents an ownership-based sharing rule in API version 33.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <sharingOwnerRules>
        <fullName>MyCase</fullName>
        <accessLevel>Edit</accessLevel>
        <description>my case test owner sharing rule desc</description>
        <label>MyCase</label>
        <sharedFrom>
            <role>COO</role>
        </sharedFrom>
        <sharedTo>
            <role>CEO</role>
        </sharedTo>
    </sharingOwnerRules>
</SharingRules>
```

The following sample XML definition represents a territory-based sharing rule in API version 33.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <sharingTerritoryRules>
        <fullName>MyAccountTerritoryRule</fullName>
        <accessLevel>Read</accessLevel>
        <accountSettings>
            <caseAccessLevel>None</caseAccessLevel>
```

```
            <contactAccessLevel>Read</contactAccessLevel>
            <opportunityAccessLevel>None</opportunityAccessLevel>
        </accountSettings>
        <description>MyAccountTerritoryRule desc</description>
        <label>MyAccountTerritoryRule</label>
        <sharedFrom>
            <territory>My_territory</territory>
        </sharedFrom>
        <sharedTo>
            <role>CEO</role>
        </sharedTo>
    </sharingTerritoryRules>
</SharingRules>
```

The following is the definition of two account owner-based sharing rules in API version 32.0 and earlier. The file name corresponds to `Account.sharingRules` under the `accountSharingRules` directory. In this definition, ownerRules corresponds to AccountOwnerSharingRule.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountSharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <ownerRules>
        <fullName>G1Dev_G2New</fullName>
        <sharedFrom>
            <group>G1Dev</group>
        </sharedFrom>
        <sharedTo>
            <group>G2New</group>
        </sharedTo>
        <accountAccessLevel>Read</caseAccessLevel>
        <caseAccessLevel>None</caseAccessLevel>
        <contactAccessLevel>Read</contactAccessLevel>
        <name>G1Dev_G2New</name>
        <opportunityAccessLevel>Edit</opportunityAccessLevel>
    </ownerRules>
    <ownerRules>
        <fullName>G2New_R1New</fullName>
        <sharedFrom>
            <group>G2New</group>
        </sharedFrom>
        <sharedTo>
            <roleAndSubordinates>R1New</roleAndSubordinates>
        </sharedTo>
        <accountAccessLevel>Edit</accountAccessLevel>
        <caseAccessLevel>Read</caseAccessLevel>
        <contactAccessLevel>Edit</contactAccessLevel>
        <name>G2New_R1New</name>
        <opportunityAccessLevel>None</opportunityAccessLevel>
    </ownerRules>
</AccountSharingRules>
```

The following is the definition of a user criteria-based sharing rule and a user membership-based sharing rule in API version 32.0 and earlier. 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>ObjA__c.*</members>
        <name>SharingCriteriaRule</name>
    </types>
    <types>
        <members>ObjA__c.*</members>
        <name>SharingOwnerRule</name>
    </types>
    <version>36.0</version>
</Package>
```

BaseSharingRule

This component is removed as of API version 33.0 and is available in earlier versions only. Use SharingBaseRule instead.

Represents the base container for criteria-based and owner-based sharing rules. This type 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 "Sharing Settings" in the Salesforce online help.

Field	Field Type	Description
`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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

CriteriaBasedSharingRule

This component is removed as of API version 33.0 and is available in earlier versions only. Use SharingRules instead.

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 Salesforce online help.

Note: You can't create a CrteriaBasedSharingRule 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 "Sharing Settings" in the Salesforce online help.

Field	Field Type	Description
`criteriaItems`	FilterItem[]	List that represents the criteria for the sharing rule. The possible values are: • `field` • `operation` • `value`

AccountCriteriaBasedSharingRule

Represents a criteria-based sharing rule for accounts. It extends the CriteriaBasedSharingRule metadata type and inherits its `criteriaItems` field.

AccountCriteriaBasedSharingRule is used by the `criteriaBasedRules` field in AccountSharingRules.

Field	Field Type	Description
`accountAccessLevel`	`ShareAccessLevelNoNone` (enumeration of type string)	Required. A value that represents the level of access that the user or group has to the account. The possible values are: • `Read` • `Edit` • `All`
`booleanFilter`	string	Represents the filter logic of the sharing rule.
`caseAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that the user or group has to cases associated with the account. The possible values are: • `None` • `Read` • `Edit`
`contactAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that the user or group has to contacts associated with the account. The possible values are: • `None` • `Read` • `Edit`
`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.
`opportunityAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a target group is granted for any associated opportunity. The possible values are: • `None` • `Read` • `Edit`

CampaignCriteriaBasedSharingRule

Represents a criteria-based sharing rule for campaigns. It extends the CriteriaBasedSharingRule metadata type and inherits its `criteriaItems` field.

CampaignCriteriaBasedSharingRule is used by the `criteriaBasedRules` field in CampaignSharingRules.

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.
`campaignAccessLevel`	`ShareAccessLevelNoNone` (enumeration of type string)	Required. A value that represents the level of access that a target group is granted for a campaign. The possible values are: • `Read` • `Edit` • `All`
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

CaseCriteriaBasedSharingRule

Represents a criteria-based sharing rule for cases. It extends the CriteriaBasedSharingRule metadata type and inherits its `criteriaItems` field.

CaseCriteriaBasedSharingRule is used by the `criteriaBasedRules` field in CaseSharingRules.

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.
`caseAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access being granted for a case. The possible values are: • `Read` • `Edit`
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

ContactCriteriaBasedSharingRule

Represents a criteria-based sharing rule for contacts. It extends the CriteriaBasedSharingRule metadata type and inherits its `criteriaItems` field.

ContactCriteriaBasedSharingRule is used by the `criteriaBasedRules` field in ContactSharingRules.

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.
`contactAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access being granted to the target group, role, or user for a contact. The possible values are: • `Read` • `Edit`
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

LeadCriteriaBasedSharingRule

Represents a criteria-based sharing rule for leads. It extends the CriteriaBasedSharingRule metadata type and inherits its `criteriaItems` field.

LeadCriteriaBasedSharingRule is used by the `criteriaBasedRules` field in LeadSharingRules.

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.
`leadAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of allowed access. The possible values are: • `Read` • `Edit`
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

OpportunityCriteriaBasedSharingRule

Represents a criteria-based sharing rule for opportunities. It extends the CriteriaBasedSharingRule metadata type and inherits its `criteriaItems` field.

OpportunityCriteriaBasedSharingRule is used by the `criteriaBasedRules` field in OpportunitySharingRules.

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.
`opportunityAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of allowed access. The possible values are: • `Read` • `Edit`
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

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
`accessLevel`	string	Required. A value that represents the type of allowed sharing. The possible values are: • `Read` • `Edit` • `All`
`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.

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: • `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>
```

```
    <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

This component is removed as of API version 33.0 and is available in earlier versions only.

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" in the Salesforce 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 "Sharing Settings" in the Salesforce 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

AccountOwnerSharingRule

Represents a sharing rule for an account with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

AccountOwnerSharingRule is used by the `ownerRules` field in AccountSharingRules.

Field	Field Type	Description
`accountAccessLevel`	`ShareAccessLevelNoNone` (enumeration of type string)	Required. A value that represents the level of access that a group or role has to the account. The possible values are: • `Read` • `Edit` • `All`
`caseAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a group or role has to cases associated with the account. The possible values are: • `None` • `Read` • `Edit`
`contactAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a group or role has to contacts associated with the account. The possible values are: • `None` • `Read` • `Edit`
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.

Field	Field Type	Description
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.
`opportunityAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for any associated opportunity. The possible values are: • `None` • `Read` • `Edit`

CampaignOwnerSharingRule

Represents a sharing rule for a campaign with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

CampaignOwnerSharingRule is used by the `ownerRules` field in CampaignSharingRules.

Field	Field Type	Description
`campaignAccessLevel`	`ShareAccessLevelNoNone` (enumeration of type string)	A value that represents the level of access that a group or role is granted for a campaign. The possible values are: • `Read` • `Edit` • `All`
`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	Name for the sharing rule. Corresponds to **Label** in the user interface.

CaseOwnerSharingRule

Represents a sharing rule for a case with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

CaseOwnerSharingRule is used by the `ownerRules` field in CaseSharingRules. All the following fields are required.

Field	Field Type	Description
`caseAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a case. The possible values are: • `Read` • `Edit`

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.

ContactOwnerSharingRule

Represents a sharing rule for a contact with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

ContactOwnerSharingRule is used by the `ownerRules` field in ContactSharingRules.

Field	Field Type	Description
`contactAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a contact. The possible values are: • `Read` • `Edit`
`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.

LeadOwnerSharingRule

Represents a sharing rule for a lead with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

LeadOwnerSharingRule is used by the `ownerRules` field in LeadSharingRules.

Field	Field Type	Description
`leadAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a lead. The possible values are: • `Read` • `Edit`
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.

Field	Field Type	Description
`name`	string	Required. Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

OpportunityOwnerSharingRule

Represents a sharing rule for an opportunity with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

OpportunityOwnerSharingRule is used by the `ownerRules` field in OpportunitySharingRules.

Field	Field Type	Description
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
`opportunityAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for an opportunity. The possible values are: • `Read` • `Edit`

AccountTerritorySharingRule

Represents a rule for sharing an account within a territory. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

AccountTerritorySharingRule is used by the `ownerRules` field in AccountTerritorySharingRules.

Field	Field Type	Description
`accountAccessLevel`	`ShareAccessLevelNoNone` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for an account territory. The possible values are: • `Read` • `Edit` • `All`
`caseAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for all child cases to an account. The possible values are: • `None` • `Read`

Field	Field Type	Description
		• `Edit`
`contactAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for all related contacts on an account. The possible values are: • `None` • `Read` • `Edit`
`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.
`opportunityAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for all opportunities associated with an account. The possible values are: • `None` • `Read` • `Edit`

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
`accessLevel`	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: • `Read` • `Edit` • `All`
`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.

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 568.

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.
`userAccessLevel`	`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: • `Read` • `Edit`

SharingSet

Represents a sharing set. A sharing set defines an access mapping that grants portal or community users access to objects that are associated with their accounts or contacts. This type extends the Metadata metadata type and inherits its `fullName` field.

For example, you can grant portal or community users access to all cases related to their account record. Similarly, you can grant portal or community users access to all cases related to a parent account that is identified on the user's account record. For more information, see "Sharing Set Overview" in the Salesforce Help.

File Suffix and Directory Location

SharingSet components have the suffix `.sharingSet` and are stored in the `sharingSets` folder.

Version

SharingSet components are available in API version 30.0 and later.

Fields

Field Name	Field Type	Description
`accessMappings`	AccessMapping[]	A list of access mappings on a sharing set.
`description`	string	The sharing set description. Limit: 255 characters.
`name`	string	Required. The unique identifier for API access. Corresponds to **Sharing Set Name** on the user interface.

Field Name	Field Type	Description
`profiles`	string[]	The profiles of users that are granted access to the target objects. Valid values are: • `Authenticated Website` • `Customer Community User` • `Customer Community Login User` • `High Volume Customer Portal User` • `Overage Authenticated Website User` • `Overage High Volume Customer Portal User`

AccessMapping

AccessMapping represents an access mapping in the sharing set, which grants access to a target object by looking up to an account or contact associated with the user.

You can grant portal users access to a target object, or to both a target object and its associated objects, such as an account and its contacts and cases.

Field Name	Field Type	Description
`accessLevel`	string	The target object access level granted to the portal user. Valid values are: • `Read` • `Edit`
`objectField`	string	A lookup to the target object, which supports standard or custom fields, or an Id. For accounts or cases associated with entitlements, use `Entitlement.Account` or `Entitlement.Case`.
`object`	string	The target object to which the portal user is gaining access, and refers to one of the following: • `Account` • `Contact` • `Case` • `ServiceContract` • `User` • Custom Objects (e.g. `ObjA__c`) Portal users gain access to all order entitlements and order items under an account to which they have access.
`userField`	string	The user's lookup to an account, contact, or a standard or custom field derived from an account or contact. Either the user or the user's manager can be used in the lookup. Valid values are: • `Account` • `Account.Field`

Field Name	Field Type	Description
		• `Contact` • `Contact.Field` • `Manager.Account` • `Manager.Contact` `Field` refers to a standard or custom field based on an account or contact.

Declarative Metadata Sample Definition

The following is an example of a SharingSet component that grants users access to all contacts whose `ReportsTo` fields match the users' contacts.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingSet xmlns="http://soap.sforce.com/2006/04/metadata">
  <accessMappings>
    <accessLevel>Read</accessLevel>
    <objectField>ReportsTo</objectField>
    <object>Contact</object>
    <userField>Contact</userField>
  </accessMappings>
  <description>User Access Mapping</description>
  <name>User</name>
  <profiles>customer community user</profiles>
</SharingSet>
```

The following is an example of a SharingSet component that grants users access to all cases that are related to an entitlement, which is associated with the user's account.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingSet xmlns="http://soap.sforce.com/2006/04/metadata">
  <name>Case</name>
   <accessMappings>
    <accessLevel>Edit</accessLevel>
    <objectField>Entitlement.Account</objectField>
    <object>Case</object>
    <userField>Account</userField>
  </accessMappings>
</SharingSet>
```

The following is an example of a SharingSet component with a list of access mappings.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingSet xmlns="http://soap.sforce.com/2006/04/metadata">
  <description>This is a basic sharing set with several access mappings.</description>
  <name>Basic</name>
  <profiles>customer community user</profiles>
  <accessMappings>
    <accessLevel>Read</accessLevel>
    <objectField>Id</objectField>
    <object>Account</object>
    <userField>Account</userField>
```

```
  </accessMappings>
  <accessMappings>
    <accessLevel>Edit</accessLevel>
    <objectField>Account</objectField>
    <object>Contact</object>
    <userField>Account</userField>
  </accessMappings>
  <accessMappings>
    <accessLevel>Edit</accessLevel>
    <objectField>Contact</objectField>
    <object>Case</object>
    <userField>Contact</userField>
  </accessMappings>
  <accessMappings>
    <accessLevel>Read</accessLevel>
    <objectField>AccountLookup__c</objectField>
    <object>HVPUAccessible__c</object>
    <userField>Account</userField>
  </accessMappings>
</SharingSet>
```

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

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <fullName>SharingSetBasic</fullName>
  <types>
    <members>HVPUAccessible__c.AccountLookup__c</members>
    <members>HVPUAccessible__c.ContactLookup__c</members>
    <name>CustomField</name>
  </types>
  <types>
    <members>HVPUAccessible__c</members>
    <name>CustomObject</name>
  </types>
  <types>
    <members>Basic</members>
    <name>SharingSet</name>
  <types>
  <version>30.0</version>
</Package>
```

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 sandbox 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 sandbox 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>
```

Skill

Represents the settings for a skill used to route chats to agents in Live Agent, such as the name of the skill and which agents the skills are assigned to. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Skill values are stored in the `<developer_name>.skill` file in the `skills` directory.

Version

Skill is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`assignments`	SkillAssignments	Specifies how skills are assigned to Live Agent users. Skills can be assigned to sets of users or sets of profiles.
`label`	string	Specifies the name of the skill.

SkillAssignments

Represents which users and user profiles to whom specific skills are assigned.

Fields

Field Name	Field Type	Description
`profiles`	SkillProfileAssignments	Specifies the profiles that are associated with a specific skill.
`users`	SkillUserAssignments	Specifies the users that are associated with a specific skill.

SkillProfileAssignments

Represents the profiles that are associated with a specific skill.

Fields

Field Name	Field Type	Description
`profile`	string	Specifies the custom name of the profile associated with a specific skill.

SkillUserAssignments

Represents the users that are associated with a specific skill.

Fields

Field Name	Field Type	Description
user	string	Specifies the username of the user associated with a specific skill.

Declarative Metadata Sample Definition

This is a sample of a `skill` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Skill xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>My Skill 1</label>
    <assignments>
        <profiles>
            <profile>LiveAgentOperator</profile>
            <profile>LiveAgentSupervisor</profile>
        </profiles>
        <users>
            <user>jdoe@acme.com</user>
        </users>
    </assignments>
</Skill>
```

StaticResource

Represents a static resource file, often a code library in a ZIP file. This metadata type extends the MetadataWithContent component and shares its fields.

Static resources allow you to upload content that you can reference in a Visualforce page, including archives (such as .zip and .jar files), images, style sheets, JavaScript, and other files.

File Suffix and Directory Location

The file suffix is `.resource` for the template file. The accompanying metadata file is named *`resource`*`-meta.xml`.

Static resource components are stored in the `staticresources` folder in the corresponding package directory.

Version

Static resources are available in API version 12.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`cacheControl`	StaticResourceCacheControl (enumeration of type string)	Required. Indicates whether the static resource is marked with a public caching tag so that a third-party delivery client can cache the content. This is a new field in API version 14.0. The valid values are: • Private • Public
`content`	base64Binary	The static resource content. 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.
`contentType`	string	Required. The content type of the file, for example text/plain.
`description`	string	The description of the static resource.
`fullName`	string	The static resource 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. 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.

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<StaticResource xmlns="http://soap.sforce.com/2006/04/metadata">
    <contentType>text/plain</contentType>
    <description>Test Resource</description>
</StaticResource>
```

SynonymDictionary

Represents a set of synonym groups, which are groups of words or phrases that are treated as equivalent in users' searches. You can define synonym groups to optimize search results for acronyms, variations of product names, and other terminology unique to your organization. Synonyms are available in Service Cloud features such as Salesforce Knowledge. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

SynonymDictionary components have the suffix `.synonymDictionary` and are stored in the `synonymDictionaries` folder.

Version

SynonymDictionary components are available in API version 29.0 and later.

Special Access Rules

Synonyms must be enabled in your organization. Only users with the "Manage Synonyms" permission can access this object.

Fields

Field Name	Field Type	Description
`groups`	SynonymGroup	The synonym groups defined in this dictionary.
`isProtected`	boolean	Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`label`	string	Required. Specifies the display name of the synonym dictionary.

SynonymGroup

Represents a group of synonymous words or phrases.

Field Name	Field Type	Description
`languages`	Language	Required. Specifies the languages the synonym group applies to. If synonyms are specific to a single language, specify only that language. If the synonyms apply to multiple languages, specify multiple languages for one synonym group.
`terms`	string	Required. A word or phrase synonymous with other terms in the group. Maximum of 50 characters. Minimum of two `terms` per group. Synonym groups are symmetric, which means that if oranges and apples are defined in a synonym group, a search for *`oranges`* will return a match for *`apples`*, and vice versa for a search for *`apples`*.

Declarative Metadata Sample Definition

The following is an example of a SynonymDictionary component:

```
<?xml version="1.0" encoding="UTF-8"?>
<SynonymDictionary xmlns="http://soap.sforce.com/2006/04/metadata">
    <groups>
        <languages>en_US</languages>
        <terms>Salesforce</terms>
        <terms>salesforce.com</terms>
        <terms>The Customer Company</terms>
        <terms>SFDC</terms>
    </groups>
    <groups>
        <languages>fr</languages>
        <terms>renault</terms>
```

```
        <terms>clio</terms>
    </groups>
    <label>Sample Dictionary</label>
</SynonymDictionary>
```

The following is an example `package.xml` that references the SynonymDictionary component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Sample Dictionary</members>
        <name>SynonymDictionary</name>
    </types>
    <version>36.0</version>
</Package>
```

Usage

If you have existing synonym groups defined before API version 29.0, your existing groups are associated with a default dictionary called `_Default`.

If you have a set of synonyms that require frequent updates, we recommend assigning the synonym group or groups to a dedicated dictionary with a small number of groups. Each time you deploy an existing dictionary, all of its synonym groups are overwritten. We don't support deploying updates to only a single synonym group within a dictionary.

Territory

Represents a territory in your organization.

Declarative Metadata File Suffix and Directory Location

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

Version

Territory components are available in API version 24.0 and later.

Fields

This metadata type extends to subtype RoleOrTerritory.

Field Name	Field Type	Description
`accountAccessLevel`	string	Specifies whether users in this territory can access accounts that are assigned to this territory and are otherwise inaccessible. Valid values are: • `Read` • `Edit` • `All`

Field Name	Field Type	Description
		If your organization's sharing model for accounts is Public Read/Write, valid values are only `Edit` and `All`. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup. This field is available in API version 31.0 and later.
`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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component. Corresponds to **Territory Name** in the user interface.
`parentTerritory`	string	The territory above this territory in the territory hierarchy.

Declarative Metadata Sample Definition

The following is the definition of a territory.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory xmlns="http://soap.sforce.com/2006/04/metadata">
    <accountAccessLevel>Edit</accountAccessLevel>
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <description>Sample Territory</description>
    <mayForecastManagerShare>false</mayForecastManagerShare>
    <name>T22name</name>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
</Territory>
```

Territory2

Represents the metadata associated with a sales territory in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your organization.

File Suffix and Directory Location

Territory2 components have the suffix `territory2` and are stored in the `territories` folder under the folder for the corresponding Territory2Model.

Version

Territory2 components are available in API version 32.0 and later.

Special Access Rules

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only territories that belong to the model in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.

Fields

Field Name	Field Type	Description
`accountAccessLevel`	string	Specifies whether users in this territory can access accounts that are assigned to this territory and are otherwise inaccessible. Valid values are: • `Read` • `Edit` • `All` If your organization's sharing model for accounts is Public Read/Write, valid values are only `Edit` and `All`. If no value is set for this field, this field value uses the default access level that is specified in Territory2Settings as permitted by the organization's sharing settings.
`caseAccessLevel`	string	Specifies whether users in this territory can access cases that are assigned to this territory and are otherwise inaccessible. Valid values are: • `None` • `Read` • `Edit` No value should be specified if your organization's sharing model for cases/opportunities is Public Read/Write,If no value is set for this field, this field value uses the default access level that is specified in Territory2Settings as permitted by the organization's sharing settings.
`contactAccessLevel`	string	Specifies whether users in this territory can access contacts that are assigned to this territory and are otherwise inaccessible. Valid values are: • `None` • `Read` • `Edit` No value should be specified if your organization's sharing model for contacts is Public Read/Write or Controlled By Parent,

Field Name	Field Type	Description
customFields	FieldValue	Values for custom fields defined on the Territory2 object and used by this territory. Their metadata is captured separately in CustomObject on page 209. Note the following: • Territory2 and Territory2Model objects do not handle values for Text Area (Long), Text Area (Rich), and text-encrypted custom fields. • Fields are referenced using their API names. Compound field types like Geolocation will appear as their constituent column fields. For example, nnn_Latitude__s, nnn_Longitude__s where "nnn" is the field name and the suffixes are the Geolocation components. • Values of required custom fields are enforced during the deploy() operation.
description	string	A description of the territory.
name	string	Required. The user interface label for the territory.
opportunityAccessLevel	string	Specifies whether users in this territory can access opportunities that are assigned to this territory and are otherwise inaccessible. Valid values are: • None • Read • Edit No value should be specified if your organization's sharing model for cases/opportunities is Public Read/Write,If no value is set for this field, this field value uses the default access level that is specified in Territory2Settings as permitted by the organization's sharing settings.
parentTerritory	string	The name of the territory's parent. When you specify the parent territory, use the developer name. Do not use the "fully qualified" name. Custom fields with no values are retrieved with values of type: <value xsi:nil="true"/>. You can also use <value xsi:nil="true"/> syntax to remove existing values in custom fields.
ruleAssociations	Territory2RuleAssociation	Represents an object assignment rule and its association to a territory. Use the developer name of the rule.
territory2Type	string	Required. The territory type that the territory belongs to.

FieldValue

Represents the values of custom fields on the Territory2 object. Available in API version 32.0 and later.

Field Name	Field Type	Description
name	string	Required. The user interface label for the territory.

Field Name	Field Type	Description
value	any type	The value of the field, which can also be null. The field type is specified in the XML and depends on the field value.

Territory2RuleAssociation

Represents the association of an object assignment rule to a territory. Available in API version 32.0 and later.

Field Name	Field Type	Description
inherited	boolean	Required. Indicates whether the rule is *inherited* from a parent territory (true) or *local* to the current territory (false). Rule inheritance flows from the parent territory where the rule is created to the rule's descendent territories (if any) in the territory model hierarchy. A local rule is created within a single territory and affects that territory only.
ruleName	string	Required. The name of a rule associated with the territory. ruleName doesn't need to be fully qualified because Metadata API assumes that the rule belongs to the same model as the territory.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2 component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2 xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <name>USA</name>
    <description>United States sales</description>
    <accountAccessLevel>Edit</accountAccessLevel>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <parentTerritory>Worldwide_Sales</parentTerritory>
    <territory2Type>Geo</territory2Type>
    <ruleAssociations>
        <ruleName>AccRule1</name>
        <inherited>True</inherited>
     </ruleAssociations>
     <ruleAssociations>
        <ruleName>AccRule2</name>
        <inherited>False</inherited>
     </ruleAssociations>
     <customFields>
        <name>Activation_DateTime__c</name>
        <value xsi:type="xsd:dateTime">2014-07-16T05:05:00.000Z</value>
     </customFields>
     <customFields>
```

```
        <name>AutoNumber__c</name>
        <value xsi:type="xsd:string">T# 000001</value>
     </customFields>
     <customFields>
        <name>DeactivationDate__c</name>
        <value xsi:type="xsd:date">2016-07-12</value>
     </customFields>
     <customFields>
        <name>External_Id__c</name>
        <value xsi:type="xsd:string">AB2345</value>
     </customFields>
     <customFields>
        <name>ManagersPhone__c</name>
        <value xsi:nil="true"/>
     </customFields>
</Territory2>
```

The following is a `package.xml` sample. *`FY13`* and *`FY14`* represent the names of territory models and demonstrate that rules can have *identical* developer names within *different* models. A wildcard character (*) in place of the model name can be used to retrieve all rules in all models in an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>FY13</members>
        <members>FY14</members>
        <name>Territory2Model</name>
    </types>

    <types>
        <members>FY13.USA</members>
        <members>FY13.Worldwide_Sales</members>
        <members>FY14.APAC</members>
        <members>FY14.USA</members>
        <name>Territory2</name>
    </types>

    <version>36.0</version>
</Package>
```

Usage

- Triggers defined on Territory2 will *not* fire during a `deploy()` operation.
- Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.

Territory2Model

Represents the metadata associated with a territory model in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your organization.

File Suffix and Directory Location

Territory2Model components have the suffix `territory2Model` and are stored in the `territory2Models` folder.

Version

Territory2Model components are available in API version 32.0 and later.

Special Access Rules

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only models in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.

Fields

Field Name	Field Type	Description
`customFields`	FieldValue	Custom fields defined on the Territory2Model object and used by this model. Their metadata is captured separately. • Territory2 and Territory2Model objects do not handle values for Text Area (Long), Text Area (Rich), and text-encrypted custom fields. • Fields are referenced using their API names. Compound field types like Geolocation will appear as their constituent column fields. For example, `nnn_Latitude__s`, `nnn_Longitude__s` where "nnn" is the field name and the suffixes are the Geolocation components. • Values of required custom fields are enforced during the `deploy()` operation.
`description`	string	A description for the territory model.
`name`	string	Required. The user interface label for the territory model..

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2Model component.

```
<?xml version="1.0" encoding="UTF-8"?>
    <Territory2Model xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <name>FY13</name>
    <description>Geographic allocation</description>
```

```
    <customFields>
        <name>Activation_DateTime__c</name>
        <value xsi:type="xsd:dateTime">2014-07-16T05:05:00.000Z</value>
    </customFields>
    <customFields>
        <name>AutoNumber__c</name>
        <value xsi:type="xsd:string">M# 000001</value>
    </customFields>
    <customFields>
        <name>DeactivationDate__c</name>
        <value xsi:type="xsd:date">2016-07-12</value>
    </customFields>
    <customFields>
        <name>External_Id__c</name>
        <value xsi:nil="true"/>
    </customFields>
</Territory2Model>
```

Usage

- The `retrieve()` call *will not* return models in these four states: `Cloning`, `Cloning Failed`, `Deleting`, and `Deletion Failed`.
- Whenever a model is created, its initial state is `Planning`. You can only do a `deploy()` operation for models in `Planning` or `Active` state. The same requirement applies to territories and rules associated with those models. For example, you might have a model in `Planning` state on a sandbox org, and a model with the same developer name in `Archived` state on your production org. The `deploy()` operation on production will fail because that model's state is `Archived` and that state prevents changes to the model.
- Because of the state restrictions, if you have territory models in different organizations with identical developer names and you attempt a `deploy()` operation, Metadata API will attempt to create new models, but that operation will fail because of the developer name conflict. For example, you might have a model in `Planning` state on a sandbox org, and a model with the same developer name in `Archived` state on your production org. The `deploy()` operation on production will fail because that model's state is `Archived` and that state prevents changes to the model.
- If you try to delete a model that has territories, then the `delete()` call will change the model's state to `Deleting` and cascade delete all territories, rules, and user associations in the model. Deleting may take some time depending on the number of territries in the model.
- Whenever a model is created, its initial state is `Planning`. If a model with the same developer name already exists, it will already have a state, so we do not include the `State` field in Territory2.
- Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.

Territory2Rule

Represents the metadata associated with a territory assignment rule associated with an object, such as Account, in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your organization.

File Suffix and Directory Location

Territory2Rule components have the suffix `territory2Rule` and are stored in the `rules` folder under the folder for the corresponding Territory2Model.

Version

Territory2Rule components are available in API version 32.0 and later.

Special Access

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only rules that belong to the model in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.The SOAP API and the user interface require that a user attempting to create or edit a rules has field-level security access to the fields referenced in the rule item. This restriction is relaxed for Metadata API `deploy()` operations, as those require "Modify All Data" and "Manage Territories" permissions. "Modify All Data" is the base permission requirement for all Metadata API operations.

Fields

Field Name	Field Type	Description
`active`	boolean	Required. Indicates whether the rule is active (`true`) or inactive (`false`). Via the API, active rules run automatically when object records are created and edited. The exception is when the value of the `IsExcludedFromRealign` field on an object record is `true`, which prevents record assignment rules from evaluating that record.
`booleanFilter`	string	An advanced filter condition. For example: `(1 AND 2) OR 3`. Numbering must start at 1 and must be contiguous.
`name`	string	Required. The user interface label for the rule.
`objectType`	string	Required. The object that the rule is defined for. For API version 32.0, the only available object is Account.
`ruleItems`	Territory2RuleItem on page 601	The items that define a rule's the selection criteria, such as `Billing State equals California`.

Territory2RuleItem

Represents the association of a rule item to a rule. Available in API version 32.0 and later.

Field Name	Field Type	Description
`field`	string	The standard or custom object field that the rule item operates on.

Field Name	Field Type	Description
operation	FilterOperation (enumeration of type string)	The criterion to apply for the rule item. For example: *equals* or *starts with*.
value	string	The field value or values to evaluate. For example: if the field is Billing ZIP/Postal Code, a value might be 94105.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2RuleItem component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2Rule xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>Northern CA</label>
    <description>To capture northern CA based accounts</description>
    <objectType>Account</objectType>
    <active>True</active>
    <ruleItems>
        <field>BillingZip</field>
        <operation>contains</operation>
        <value><94105,94404,94536/value>
    </ruleItems>
    <ruleItems>
        <field>Industry</field>
        <operation>equals</operation>
        <value>IT</value>
    </ruleItems>
    <ruleItems>
        <field>someCustomField__c</field>
        <operation>greater_than</operation>
        <value>50000</value>
    </ruleItems>
    <booleanFilter>(1 OR 2) AND 3</booleanFilter>
</Territory2Rule>
```

The following is a package.xml sample. *FY13* and *FY14* represent names of territory models and demonstrate that rules can have *identical* developer names within *different* models. A wildcard character (*) in place of the model name can be used to retrieve all rules in all models in an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
        <types>
            <members>FY13</members>
            <members>FY14</members>
            <name>Territory2Model</name>
        </types>

        <types>
            <members>FY13.AccRule1</members>
```

```
            <members>FY14.AccRule1</members>
            <name>Territory2Rule</name>
        </types>

        <version>36.0</version>
</Package>
```

Usage

- A territory rule can have up to 10 rule items.
- The sort order of rule items is implicitly derived from the position of the rule items in the XML
- Rules can't be run via Metadata API.
- Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.

Territory2Type

Represents the metadata for a category of territories in Territory Management 2.0. Every Territory2 must have a Territory2Type. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Enterprise Territory Management has been enabled for your organization.

File Suffix and Directory Location

Territory2Type components have the suffix `territory2Type` and are stored in the `territory2Types` folder.

Version

Territory2Type components are available in API version 32.0 and later.

Special Access Rules

Users without the "Manage Territories" permission will be able to retrieve all the Territory2Types in the organization. "Manage Territories" permission is required for the `deploy()` operation, in addition to the "Modify All Data" permission required by the Metadata API.

Fields

Field Name	Field Type	Description
`description`	string	A description of the territory type.
`name`	string	Required. The user interface label for the territory type.
`priority`	int	Required. Used for Filter-Based Opportunity Territory Assignment (Pilot in Spring '15 / Metadata API version 33). Lets you specify a priority for a territory type. For opportunity assignments, the filter examines all territories assigned to the account that the opportunity is assigned to. The account-assigned territory whose territory type priority is highest is then assigned to the opportunity. The

Field Name	Field Type	Description
		`priority` field value on each territory type must be unique. Further, if there are multiple territories with the same territory type (and therefore the same priority) assigned to the account, no territory is not assigned to the opportunity.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2Type component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2Type xmlns="http://soap.sforce.com/2006/04/metadata">
    <name>Geo</name>
    <description>Geographic allocation</description>
</Territory2Type>
```

Usage

Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.

TransactionSecurityPolicy

Represents a transaction security policy definition. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

TransactionSecurityPolicy components have the suffix `.transactionSecurityPolicy` and are stored in the `transactionSecurityPolicies` folder.

Version

TransactionSecurityPolicy components are available in API version 35.0 and later.

Fields

Field Name	Field Type	Description
`action`	TransactionSecurityAction	Required. Describes the action to take when the matching Transaction Security policy is triggered.
`active`	boolean	Required. If `true`, the policy is enabled and is actively monitoring its event.
`apexClass`	string	Required. The name of the class that implements the `TxnSecurity.PolicyCondition` interface for this policy.

Field Name	Field Type	Description
`eventType`	MonitoredEvents (enumeration of type string)	Indicates which type of event is being monitored. Valid values are: • `AccessResource`—Notifies you when the selected resource has been accessed. • `AuditTrail`—Reserved for future use. • `DataExport`—Notifies you when the selected object type has been exported using the Data Loader API client. • `Entity`—Notifies you on use of an object type such as an authentication provider or client browser. • `Login`—Notifies you when a user logs in.
`executionUser`	string	Required. The name of the user to notify when the policy is triggered, if any notifications have been selected. This user must have the System Administrator profile.
`resourceName`	string	Required. A resource used to narrow down the conditions under which the policy triggers. For example, with a `Login` event, you can add a resource to specify that only a specific login URL triggers the policy. The resources available depend on the `Event Type` field. Valid resources are grouped below by event type. • AccessResource—EventTimestamp, SessionLevel, SourceIp • DataExport—EventTimestamp, SessionLevel, SourceIp • Entity—AuthorizeUrl, ConsumerKey, ConsumerSecret, DefaultScopes, DeveloperName, ErrorUrl, FriendlyName, IconUrl, IdTokenIssuer, LogoutUrl, TokenUrl, UserInfoUrl • Login—ApiType, ApiVersion, Application, Browser, ClientVersion, LoginUrl, Platform, Status

TransactionSecurityAction

Describes the action to take when the matching Transaction Security policy is triggered.

Field Name	Field Type	Description
`block`	boolean	Required. If `true`, the requested operation is blocked. This action only applies to Login and AccessResource events.
`endSession`	boolean	Required. If `true`, a current session must be closed before a new session can be started. This action only applies to Login events.
`notifications`	TransactionSecurityNotification[]	Specifies how to notify the ystem administrator when the action is triggered. There can be none, one, or multiple notifications.
`twoFactorAuthentication`	boolean	Required. If `true`, two-factor authentication is required for a higher level of access before the requested operation can continue. This action only applies to Login and AccessResource events.

TransactionSecurityNotification

Describes who to notify and how to notify them when the matching Transaction Security policy is triggered.

Field Name	Field Type	Description
`inApp`	boolean	True if an in-app notification is selected.
`sendEmail`	boolean	True if an email notification is selected.
`user`	string	The administrator to receive the notification. This user must belong to the System Administrator profile.

Declarative Metadata Sample Definition

The following is an example of a TransactionSecurityPolicy component.

```
<?xml version="1.0" encoding="UTF-8"?>
<TransactionSecurityPolicy xmlns="http://soap.sforce.com/2006/04/metadata">
    <action>
        <block>true</block>
        <endSession>false</endSession>
        <notifications>
            <inApp>false</inApp>
            <sendEmail>true</sendEmail>
            <user>admin@your.org</user>
        </notifications>
        <twoFactorAuthentication>false</twoFactorAuthentication>
    </action>
    <active>true</active>
    <apexClass>TxnSecurityMdApiPolicy</apexClass>
    <eventType>Login</eventType>
    <executionUser>admin@your.org</executionUser>
    <resourceName>LoginHistory</resourceName>
</TransactionSecurityPolicy>
```

The following is an example package manifest used to deploy or retrieve the transaction security metadata for an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MySecurityPolicy</members>
        <name>TransactionSecurityPolicy</name>
    </types>
    <version>35.0</version>
</Package>
```

Translations

This metadata type allows you to work with translations for various supported languages. This type extends the Metadata metadata type and inherits its `fullName` field. The ability to translate component labels is part of the Translation Workbench. For more information, see “Enable and Disable the Translation Workbench” in the Salesforce online help.

Language

Salesforce offers three levels of language support: fully supported languages, end-user languages, and platform-only languages. Each language is identified by a two-character language code, such as `en`, or a five-character locale code, such as en_AU.

Note: Setting a default locale is different from setting a default language.

Salesforce offers full support for the following languages.

- Chinese (Simplified): `zh_CN`
- Chinese (Traditional): `zh_TW`
- Danish: `da`
- Dutch: `nl_NL`
- English: `en_US`
- Finnish: `fi`
- French: `fr`
- German: `de`
- Italian: `it`
- Japanese: `ja`
- Korean: `ko`
- Norwegian: `no`
- Portuguese (Brazil): `pt_BR`
- Russian: `ru`
- Spanish: `es`
- Spanish (Mexico): `es_MX`
- Swedish: `sv`
- Thai: `th`

End-user languages are useful if you have a multilingual organization or partners who speak languages other than your company's default language. For end-user languages, Salesforce provides translated labels for all standard objects and pages, *except* administrative pages, Setup, and Help. When you specify an end-user language, labels and Help that aren't translated appear in English. End-user languages are intended only for personal use by end users. Don't use end-user languages as corporate languages. Salesforce doesn't provide customer support in end-user languages.

End-user languages include:

- Arabic: `ar`
- Bulgarian: `bg`
- Croatian: `hr`
- Czech: `cs`
- English (UK): `en_GB`
- Greek: `el`
- Hebrew: `iw`
- Hungarian: `hu`
- Indonesian: `in`
- Polish: `pl`

- Portuguese (Portugal): `pt_PT`
- Romanian: `ro`
- Slovak: `sk`
- Slovenian: `sl`
- Turkish: `tr`
- Ukrainian: `uk`
- Vietnamese: `vi`

Note: Salesforce provides limited support for right-to-left languages—Arabic and Hebrew—for the following features.

- Live Agent
- Cases
- Accounts

These features are not supported in Lightning Experience, the Salesforce1 mobile app, any other mobile app or mobile browser, or any user interface except Salesforce Classic. There is no guarantee that right-to-left languages function correctly with any other Salesforce features. There are no plans to expand the list of supported features.

Features that aren't supported for right-to-left languages include, but are not limited to, the following.

- Report Builder
- Generating quote PDFs
- Customizable forecasting
- Emails
- Salesforce Knowledge
- Feeds
- Communities

The absence of a feature from this list does not imply support. Only Live Agent, Cases, and Accounts are supported with right-to-left languages.

In situations where Salesforce doesn't provide default translations, use platform-only languages to localize apps and custom functionality that you've built on the Salesforce App Cloud. You can translate items such as custom labels, custom objects, and field names. You can also rename most standard objects, labels, and fields. Informative text and non-field label text aren't translatable.

Platform-only languages are available in all places where you can select a language in the application. However, when you select a platform-only language, all standard Salesforce labels default to English or, in select cases, to an end-user or fully supported language.

- Albanian: `sq`
- Arabic (Algeria): `ar_DZ`
- Arabic (Bahrain): `ar_BH`
- Arabic (Egypt): `ar_EG`
- Arabic (Iraq): `ar_IQ`
- Arabic (Jordan): `ar_JO`
- Arabic (Kuwait): `ar_KW`
- Arabic (Lebanon): `ar_LB`
- Arabic (Libya): `ar_LY`
- Arabic (Morocco): `ar_MA`
- Arabic (Oman): `ar_OM`

- Arabic (Qatar): `ar_QA`
- Arabic (Saudi Arabia): `ar_SA`
- Arabic (Sudan): `ar_SD`
- Arabic (Syria): `ar_SY`
- Arabic (Tunisia): `ar_TN`
- Arabic (United Arab Emirates): `ar_AE`
- Arabic (Yemen): `ar_YE`
- Armenian: `hy`
- Basque: `eu`
- Bosnian: `bs`
- Bengali: `bn`
- Chinese (Simplified—Singapore): `zh_SG`
- Chinese (Traditional—Hong Kong): `zh_HK`
- English (Australia): `en_AU`
- English (Canada): `en_CA`
- English (Hong Kong): `en_HK`
- English (India): `en_IN`
- English (Ireland): `en_IE`
- English (Malaysia): `en_MY`
- English (Philippines): `en_PH`
- English (Singapore): `en_SG`
- English (South Africa): `en_ZA`
- Estonian: `et`
- French (Belgium): `fr_BE`
- French (Canada): `fr_CA`
- French (Luxembourg): `fr_LU`
- French (Switzerland): `fr_CH`
- Georgian: `ka`
- German (Austria): `de_AT`
- German (Luxembourg): `de_LU`
- German (Switzerland): `de_CH`
- Hindi: `hi`
- Icelandic: `is`
- Irish: `ga`
- Italian (Switzerland): `it_CH`
- Latvian: `lv`
- Lithuanian: `lt`
- Luxembourgish: `lb`
- Macedonian: `mk`
- Malay: `ms`

- Maltese: `mt`
- Romanian (Moldova): `ro_MD`
- Montenegrin: `sh_ME`
- Romansh: `rm`
- Serbian (Cyrillic): `sr`
- Serbian (Latin): `sh`
- Spanish (Argentina): `es_AR`
- Spanish (Bolivia): `es_BO`
- Spanish (Chile): `es_CL`
- Spanish (Colombia): `es_CO`
- Spanish (Costa Rica): `es_CR`
- Spanish (Dominican Republic): `es_DO`
- Spanish (Ecuador): `es_EC`
- Spanish (El Salvador): `es_SV`
- Spanish (Guatemala): `es_GT`
- Spanish (Honduras): `es_HN`
- Spanish (Nicaragua): `es_NI`
- Spanish (Panama): `es_PA`
- Spanish (Paraguay): `es_PY`
- Spanish (Peru): `es_PE`
- Spanish (Puerto Rico): `es_PR`
- Spanish (United States): `es_US`
- Spanish (Uruguay): `es_UY`
- Spanish (Venezuela): `es_VE`
- Tagalog: `tl`
- Tamil: `ta`
- Urdu: `ur`
- Welsh: `cy`

Declarative Metadata File Suffix and Directory Location

Translations are stored in a file with a format of *`localeCode`*`.translation`, where *`localeCode`* is the locale code of the translation language. For example, the file name for German translations is `de.translation`. The supported locale codes are listed in Language.

Custom object translations are stored in the `translations` folder in the corresponding package directory.

Version

Translations components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
customApplications	CustomApplicationTranslation[]	A list of custom application translations.
customLabels	CustomLabelTranslation[]	A list of custom label translations.
customPageWebLinks	CustomPageWebLinkTranslation[]	A list of translations for web links defined in a home page component.
customTabs	CustomTabTranslation[]	A list of custom tab translations.
fullName	string	Required. The language code; for example, `de` for German. 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.
quickActions	GlobalQuickActionTranslation[]	A list of global (rather than object-specific) quick actions.
reportTypes	ReportTypeTranslation[]	A list of report type translations.
scontrols	ScontrolTranslation[]	A list of s-control translations.

CustomApplicationTranslation

CustomApplicationTranslation contains details for a custom application translation. For more details, see CustomApplication.

Field	Field Type	Description
label	string	Required. The translated custom application name. Maximum of 765 characters.
name	string	Required. The name of the custom application.

CustomLabelTranslation

CustomLabelTranslation contains details for a custom label translation. For more details, see CustomLabels.

Field	Field Type	Description
label	string	Required. The translated custom label name. Maximum of 765 characters.
name	string	Required. The custom label name.

CustomPageWebLinkTranslation

CustomPageWebLinkTranslation contains details for a translation of a web link defined in a home page component. For more details, see CustomPageWebLink.

Field	Field Type	Description
label	string	Required. The translated web link.
name	string	Required. The name of the web link.

CustomTabTranslation

CustomTabTranslation contains details for a translation of a custom tab. For more details, see CustomTab.

Field	Field Type	Description
label	string	Required. The translated custom tab name.
name	string	Required. The custom tab name.

GlobalQuickActionTranslation

GlobalQuickActionTranslation contains details for the translation of a quick action, globally. For more information, see QuickAction.

Field	Field Type	Description
label	string	Required. The translated quick action name, globally.
name	string	Required. The quick action name.

ReportTypeTranslation

ReportTypeTranslation contains details for a translation of a custom report type. For more details, see ReportType.

Field	Field Type	Description
description	string	The translated report type description.
label	string	The translated report type name.
name	string	Required. The name of the report type.
sections	ReportTypeSectionTranslation[]	A list of report type section translations.

ReportTypeSectionTranslation

ReportTypeSectionTranslation contains details for a report type section translation.

Field	Field Type	Description
columns	ReportTypeColumnTranslation[]	A list of report type column translations.
label	string	The translated report type section name.
name	string	Required. The name of the report type section.

ReportTypeColumnTranslation

ReportTypeColumnTranslation contains details for a report type column translation.

Field	Field Type	Description
label	string	Required. The translated report type column name.
name	string	Required. The report type column name.

ScontrolTranslation

Important: Visualforce pages supersede s-controls. Organizations that haven't previously used s-controls can't create them. Existing s-controls are unaffected, and can still be edited.

ScontrolTranslation contains details for a translation of an s-control. For more information, see "About S-Controls" in the Salesforce online help.

Field	Field Type	Description
label	string	Required. The translated s-control name.
name	string	Required. The name of the s-control.

Declarative Metadata Sample Definition

A sample XML definition of a translations component is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Translations xmlns="http://soap.sforce.com/2006/04/metadata">
    <customApplications>
    <label>Angebot-Manager</label>
        <name>Quote Manager</name>
    </customApplications>
    <customLabels>
    <label>Dieses ist ein manuelles Angebot</label>
        <name>quoteManual</name>
    </customLabels>
</Translations>
```

Usage

When you use the `retrieve()` call to get translations in your organization, the files returned in the `.translations` folder only include translations for the other metadata types referenced in `package.xml`. For example, the following `package.xml` file contains `types` elements that match all custom applications, custom labels, Web links defined in home page components, custom tabs, report types, and s-controls. Translations for all these metadata types are returned because each metadata type is explicitly listed in `package.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
       <members>*</members>
       <name>CustomApplication</name>
    </types>
    <types>
       <members>*</members>
       <name>CustomLabels</name>
    </types>
    <types>
       <members>*</members>
       <name>CustomPageWebLink</name>
    </types>
    <types>
       <members>*</members>
       <name>CustomTab</name>
    </types>
    <types>
       <members>*</members>
       <name>ReportType</name>
    </types>
    <types>
       <members>*</members>
       <name>Scontrol</name>
    </types>
    <types>
       <members>*</members>
        <name>Translations</name>
    </types>
    <version>36.0</version>
</Package>
```

SEE ALSO:

CustomLabels

WaveApplication (Pilot)

This metadata represents the Wave Analytics Application.

Represents the Wave Application.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

WaveApplication components have the suffix `.wapp` and are stored in the `wave` folder.

Version

WaveApplication components are available in API version 36.0 and later.

Special Access Rules

The WavePackaging permission is required to access this type.

Fields

Field Name	Field Type	Description
`assetIcon`	string	The icon that represents the Wave application.
`description`	string	The description that appears in the user interface.
`folder`	string	The internal api name of the folder or application.
`masterLabel`	string	The user interface label name of the folder or application.
`shares`	FolderShare	The folder sharing rules.

Declarative Metadata Sample Definition

The following is an example of a WaveApplication component.

```
<?xml version="1.0" encoding="UTF-8"?>
  <WaveApplication xmlns="http://soap.sforce.com/2006/04/metadata">
  <assetIcon>/analytics/wave/web/proto/images/app/icons/11.png</assetIcon>
  <description>Application that shows my sales</description>
  <folder>edit</folder>
  <masterLabel>Sales Application</masterLabel>
  <shares>
   <accessLevel>EditAllContents</accessLevel>
   <sharedTo>shareswith@org.ee</sharedTo>
   <sharedToType>User</sharedToType>
  </shares>
 </WaveApplication>
```

WaveDataset (Pilot)

This metadata reperesents the WaveDataset object in the Wave Analytics Application.

Represents the WaveDataset object in the Wave Analytics Application.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

WaveDataset components have the suffix `.wds` and are stored in the `wave` folder.

Version

WaveDataset components are available in API version 36.0 and later.

Special Access Rules

The WavePackaging permission is required to access this type.

Fields

Field Name	Field Type	Description
`application`	string	The internal name of the application.
`description`	string	The Dataset description that appears in the user interface.
`masterLabel`	string	The user interface label name of the Dataset.

Declarative Metadata Sample Definition

The following is an example of a WaveDataset component.

```
<WaveDataset>
    <application>SharedApp</application>
    <description>description</description>
    <masterLabel>l</masterLabel>
</WaveDataset>
```

WaveTemplateBundle

Represents a Wave Analytics template bundle, which can be used to create Wave apps. A bundle contains a Wave template definition and all its related resources.This type extends the Metadata metadata type and inherits its `fullName` field.

Note: We provide this feature to selected customers through a pilot program that requires agreement to specific terms and conditions. To be nominated to participate in the program, contact Salesforce. Because pilot programs are subject to change, we can't guarantee acceptance. This pilot feature isn't generally available, as referenced in this document or in press releases or public statements. We can't guarantee that it will become generally available within any particular time frame or at all. Make your purchase decisions only on the basis of generally available features. Services provided by the Wave REST API are subject to change. Support is not provided.

File Suffix and Directory Location

A Wave template bundle is a folder that contains definition files for a template. Unlike other metadata components, a WaveTemplateBundle component isn't represented by a single component file, but instead by a collection of JSON and CSV definition files. Each definition file

represents a resource in a template, such as lenses, dashboards, dataflows, and comma-separated values. For example, this directory structure shows the hierarchy of the folders and files for one Wave Template definition, template1.

```
waveTemplates
    template1
        template-info.json
        variables.json
        ui.json
        extFiles
            PostalCodes.csv
```

Wave template bundles must be under a top-level folder that's named `waveTemplates`. Each bundle must have its own subfolder under the `waveTemplates` folder and be named with the template's fully qualified API name. The bundle folder must contain a template-info.json file to specify the metadata about the template and the references to other definition files. An entire bundle doesn't have a suffix and definition files can have one of the following suffixes.

Suffix	Component Type
.json	JavaScript Object Notation
.csv	Comma-Separated Values

Version

WaveTemplateBundle components are available in API version 35.0 and later.

Special Access Rules

Definitions can be created in both managed and unmanaged packages.

Fields

Field Name	Field Type	Description
`assetIcon`	string	The icon to use by default for new Wave apps based on this template. Valid values are `1.png` through `20.png`.
`description`	string	The specification of the template.
`label`	string	Required. The label of the template.
`templateType`	string	Required. The type of the template. Valid values are: • `App` • `Dashboard` • `Lens`

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" in the Salesforce Help. This type 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 standard or 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 standard or custom object.

Field Name	Field Type	Description
`alerts`	WorkflowAlert[]	An array of all alerts for the object associated with the workflow.
`fieldUpdates`	WorkflowFieldUpdate[]	An array of all field updates for the object associated with the workflow.
`flowActions`	WorkflowFlowAction[]	An array of flow triggers for the object associated with the workflow. Available in API version 30.0 and later. The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren't available for new organizations.
`fullName`	string	The developer name used as a 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

Field Name	Field Type	Description
knowledgePublishes	WorkflowKnowledgePublish[]	An array of Salesforce Knowledge Workflow Publishes associated with the workflow. Available in API version 27.0 and later.
outboundMessages	WorkflowOutboundMessage[]	An array of all of the outbound messages for the object associated with the workflow.
rules	WorkflowRule[]	An array of all the objects associated with the workflow.
tasks	WorkflowTask[]	An array of all the tasks for the object associated with the workflow.

WorkflowActionReference

WorkflowActionReference represents one of the workflow actions.

Field Name	Field Type	Description
name	string	Required. The name of the workflow action.
type	WorkflowActionType (enumeration of type string)	Required. Available types of workflow actions: • Alert • FieldUpdate • FlowAction—Available in API version 30.0 and later • OutboundMessage • Task The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren't available for new organizations.

WorkflowAlert

WorkflowAlert represents an email alert associated with a workflow rule.

Field Name	Field Type	Description
ccEmails	string[]	Additional email addresses. This field is similar to the CC field in email clients. For the email to be sent successfully, set a value for ccEmails or recipients. You can set values for both fields. The value of ccEmails can include up to five different email addresses.
description	string	Required. A description of the email alert. Available in API version 16.0 and later.

Field Name	Field Type	Description
`fullName`	string	Required. The developer name used as a 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`recipients`	WorkflowEmailRecipient[]	The recipients for the email. For the email to be sent successfully, set a value for `ccEmails` or `recipients`. You can set values for both fields.
`senderAddress`	string	The address in the From field for the email alert. This allows you to use a standard global email address for your organization (such as `support@company.com`) instead of the default From field, which is the email address of the person who updates the record. You can only specify a value in this field if the `senderType` is set to `OrgWideEmailAddress`. See "Organization-Wide Addresses" in the Salesforce Help.
`senderType`	ActionEmailSenderType (enumeration of type string)	The email used as the sender's From and Reply-To addresses. The following values are valid: • `CurrentUser`—The email address of the person updating the record. This is the default setting. • `DefaultWorkflowUser`—The email address of the default workflow user. • `OrgWideEmailAddress`—A verified global email address for your organization, such as `support@company.com`.
`template`	string	Required. Named reference to an EmailTemplate. This email template does not have to exist in the zip file, but it must exist in Metadata API.

WorkflowEmailRecipient

WorkflowEmailRecipient represents a recipient for an email alert associated with a workflow rule.

Field Name	Field Type	Description
`field`	string	Name of the field referenced in `type`. The field named should be of the type specified in `type`.
`recipient`	string	The recipients for the email. Depending on the type selected, this may be required.
`type`	ActionEmailRecipientTypes (enumeration of type string)	Named reference to an EmailTemplate component. Valid values are: • `accountOwner` - The email is sent to the record's account owner (for example, the Account owner for an Opportunity). • `accountTeam` - Only applicable on the Account object. The email is sent to everyone on that Account's account team. • `campaignMemberDerivedOwner` - Emails are sent to lead and contact owners when contacts are added to a campaign or in response to a campaign. • `contactLookup` - The email is sent to a contact whose value is looked up from a field on the record. For this value, the `field` field must reference a Contact. • `creator` - The email is sent to the record's creator. • `customerPortalOwner` - The email is sent to a specific self-service portal user. For this value, the recipient field must reference a User (by username), only self-service portal users. • `email` - The email is sent to an email address whose value is looked up from a field on the record. For this value, the `field` field must reference an email field. • `group` - The email is sent to all users in a group. For this value, the recipient field must reference a group (by group name). • `opportunityTeam` - Only applicable on the Opportunity object. The email is sent to everyone on that Opportunity's opportunity team. • `owner` - The email is sent to the record's owner. • `partnerUser` - The email is sent to a specific partner user. For this value, the recipient field must reference a User (by username), only partner users. • `portalRole` - Like `role`, but for portal roles only. • `portalRoleSubordinates` - Like `roleSubordinates`, but for portal roles only. • `role` - The email is sent to all users in a role. For this value, the recipient field must reference a Role (in the role hierarchy, by role name).

Field Name	Field Type	Description
		• `roleSubordinates` - The email is sent to all users in a role subordinates. For this value, the recipient field must reference a Role. • `roleSubordinatesInternal` - Like `roleSubordinates`, but for internal portal roles only. • `user` - The email is sent to a specific user. For this value, the recipient field must reference a User (by username). • `userLookup` - The email is sent to a user whose value is looked up from a field on the record. For this value, the `field` field must reference a user foreign key field.

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.

Field Name	Field Type	Description
`description`	string	The description of the field update. This information is useful to track the reasoning for initially configuring the field update.
`field`	string	Required. The field (on the object for the workflow) to be updated.
`formula`	string	If the `operation` field value is `Formula`, 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 `fullName` 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.
`literalValue`	string	If the `operation` field value is `Literal`, this is the literal value for the field.
`lookupValue`	string	If the `operation` field value is `lookupValue`, this is the lookup value that is referenced.
`lookupValueType`	LookupValueType (enumeration of type string)	The type of object that the `lookupValue` field value is referencing. The valid values are: • `Queue` • `RecordType` • `User`
`name`	string	Required. A name for the component. Available in version API 16.0 and later.

Field Name	Field Type	Description
`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: • `Formula` - Indicates the field will be set to a formula. If set, the formula must be a valid formula. • `Literal` - Indicates the field will be set to a literal value. If set, the literalValue must be a valid literal value for this field. • `LookupValue` - Similar to Literal, but for an object reference, such as a contact, user, account, etc. If set, the `lookupValue` element must be set. Only User is supported in the current API. • `NextValue` - Indicates that the field will be set to its next value; this is only allowed when the field update references a picklist. • `Null` - Indicates the field will be set to null. • `PreviousValue` - 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 (`true`) or not (`false`). 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. 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, `EmailMessage.ParentId`) pointing to the parent (for example, Case). When set, the formula is based on the child object (for example, `EmailMessage`). This field is named `sourceField` before version 14.0. The field name change is automatically handled between versions and does not require any manual editing of existing XML component files.

WorkflowFlowAction

Represents a flow trigger, which is a workflow action that launches a flow. Available in API version 30.0 and later. For more information, see the following topics in the Salesforce Help.

- Define a Flow Trigger for Workflow (Pilot)
- Flow Trigger Considerations (Pilot)

Note:

- The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren't available for new organizations.
- Test mode for flow triggers isn't supported in the Metadata API. If you want a flow trigger to run the latest flow version when an administrator causes the workflow rule to fire, enable test mode via the user interface after deployment.

Field Name	Field Type	Description
`description`	string	Describes the flow trigger.
`flow`	string	Required. Unique name of the flow that this workflow action launches.
`flowInputs`	WorkflowFlowActionParameter[]	An array of values to pass into flow variables and sObject variables when launching the flow.
`label`	string	Required. Name of the flow trigger.
`language`	string	Reserved for future use.
`protected`	boolean	Reserved for future use.

WorkflowFlowActionParameter

Represents a value specified in the flow trigger that is passed into a flow variable or sObject variable when launching the flow.

Note: The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren't available for new organizations.

Field Name	Field Type	Description
`name`	string	Required. Unique name of the variable or sObject variable in the flow. The flow variable must have `isInput` set to `True`.
`value`	string	Required. Value to assign to the flow variable or sObject variable when launching the flow. Unlike an sObject variable, which represents an entire Salesforce record in the flow, a flow variable represents a single field. Therefore, the allowed `value` depends on whether the `name` identifies a flow variable or an sObject variable. For an sObject variable, the `value` must be a merge field that identifies a record—or a lookup relationship field that references a record—of the same object type as the sObject variable. For example:

Field Name	Field Type	Description
		• *{!this}*—identifies the record that fired the workflow rule. • *{!Contact}*—identifies the contact associated with the record that fired the workflow rule. • *{!Asset.Account}*—identifies the account associated with the asset that is associated with the record that fired the workflow rule. • *{!SomeObject__r}*—uses a lookup relationship field to identify a custom object record associated with the record that fired the workflow rule. For a flow variable, you can enter a merge field or a literal value. Manually enter a literal value when the variable should have the same value every time the flow is launched, regardless of which record fired the workflow rule. For example, you can enter *true* or *false* for a variable of type Boolean. For a flow variable, supported merge fields identify a global variable or a field of the same data type as the flow variable. For example: • *{!Id}*—ID of the record that fired the workflow rule. • *{!Account.Owner.Email}*—email address of the account owner for the account associated with the record that fired the workflow rule. • *{!$Organization.Country}*—country of the organization.

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: • `PublishAsNew`: Publishes the article as a new article. • `Publish`: 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 (`true`) or not (`false`). 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 and approval 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 "Outbound Message Actions" in the Salesforce 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. This API version is used in API calls back to Salesforce using the enterprise or partner WSDLs. The `API Version` can only be modified by using the Metadata API. It can't be modified using the Salesforce user interface. This field is available in API version 18.0 and later. Warning: If you change the `apiVersion` to a version that doesn't support one of the `fields` 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 entering *`Outbound Messages`* in the `Quick Find` box, then selecting **Outbound Messages** in Salesforce.
`description`	string	Describes the outbound message.
`endpointUrl`	string	Required. The endpoint URL to which the outbound message is sent.
`fields`	string[]	The named references to the fields that are to be sent.
`fullName`	string	Required. The developer name used as a 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 end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`includeSessionId`	boolean	Required. Set if you want the Salesforce *session ID* included in the outbound message. Useful if you intend to make API calls and you do not want to include a username and password.
`integrationUser`	string	Required. The named reference to the user under which this message is sent.
`name`	string	Required. A name for the component. Available in version API 16.0 and later.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.

Field Name	Field Type	Description
`useDeadLetterQueue`	boolean	This field is only available for organizations with dead letter queue permissions turned on. If set, this outbound message will use the dead letter queue if normal delivery fails.

WorkflowRule

This metadata type represents a workflow rule. This type extends the Metadata metadata type and inherits its `fullName` field.

Field Name	Field Type	Description
`actions`	WorkflowActionReference[]	An array of references for the actions that should happen when this rule fires.
`active`	boolean	Required. Determines if this rule is active.
`booleanFilter`	string	For advanced criteria filter, the boolean formula, for example, `(1 AND 2) OR 3`.
`criteriaItems`	FilterItem[]	An array of the boolean criteria (conditions) under which this rule fires. Note that either this or `formula` must be set.
`description`	string	The description of the workflow rule
`formula`	string	The formula condition under which this rule first (either this or `criteriaItems`) must be set
`fullName`	string	The developer name used as a 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 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: • `onAllChanges` - The workflow rule is considered on all changes. • `onCreateOnly` - The workflow rule is considered only on create. • `onCreateOrTriggeringUpdate` - The workflow rule is considered on create and triggering updates.

Field Name	Field Type	Description
workflowTimeTriggers	WorkflowTimeTrigger	Represents a set of Workflow actions (Field Updates, Email Alerts, Outbound Messages, and Tasks) that should execute before or after a specified interval of time.

WorkflowTask

This metadata type references an assigned workflow task.

Field Name	Field Type	Description
assignedTo	string	Specifies the user, role, or team to which the workflow rule or action is assigned. The field corresponding to the value specified here must be the same as the specified assignedToType.
assignedToType	ActionTaskAssignedToTypes (enumeration of type string)	Valid string values for this type are: • accountCreator - When set, the task is assigned to the record's account's creator. • accountOwner - When set, the task is assigned to the record's account's owner (Opportunity). • accountTeam - Same as WorkflowAlert type • creator - When set, the task is assigned to the record's creator. • opportunityTeam - Same as WorkflowAlert type • owner - When set, the task is assigned to the record's owner. • partnerUser - When set, the assignedTo field references a User (by username), a partner user. • portalRole - When set, the assignedTo field references a Role (by role name), a portal role. • role - When set, the assignedTo field references a Role (by role name) • user - When set, the assignedTo field references a User (by username)
description	string	The description of this workflow task.
dueDateOffset	int	Required. The offset, in days, from either the trigger date, or the date specified in the (optional) offsetFromField. This can be a negative number.
fullName	string	Required. The developer name used as a unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be

Field Name	Field Type	Description
		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.
`notifyAssignee`	boolean	Required. Set to send an email notification when the task is assigned.
`offsetFromField`	string	Optional field reference of the date field from which the `dueDate` should be computed.
`priority`	string	Required. The priority to assign the created task.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`status`	string	Required. The status to assign the created task.
`subject`	string	Required. A subject for the workflow task. It is used if an email notification is sent when the task is assigned. Available in API version 16.0 and later.

WorkflowTimeTrigger

Represents a set of Workflow actions (Field Updates, Email Alerts, Outbound Messages, and Tasks) 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. `Created Date`, `Last Modified Date`, `Rule Trigger Date` 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: • `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>
        <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</name>
                <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>
        <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>
```

CHAPTER 11 Headers

Use headers in Metadata API calls to set options for each call.

IN THIS SECTION:

`AllOrNoneHeader`

Indicates whether to roll back all metadata changes when some of the records in a call result in failures.

Version

This header is available in API version 34.0 and later.

Supported Calls

createMetadata(), updateMetadata(), upsertMetadata(), deleteMetadata()

Usage

If this header isn't used in API version 34.0 and later, by default a call can save a partial set of records (equivalent to `AllOrNoneHeader=false`)—the records that are processed successfully are saved and records that have failures aren't saved.

Fields

Field Name	Type	Description
`allOrNone`	boolean	Set to `true` to cause all metadata changes to be rolled back if any records in the call cause failures. Set to `false` to enable saving only the records that are processed successfully when other records in the call cause failures.

Sample Code—Java

Add the `AllOrNoneHeader` to the metadata connection before you perform a call as follows:

```
metadataConnection.setAllOrNoneHeader(true);
```

This next example shows how to use the `AllOrNoneHeader` when creating two custom objects. Because the second custom object doesn't have the required `Name` field, the `create()` call can't create this custom object and rolls back the first custom object. The output is shown after this code sample.

```
import com.sforce.soap.metadata.*;
import com.sforce.soap.metadata.Error;
import com.sforce.ws.ConnectionException;

public class CallWithHeader {

    MetadataConnection metadataConnection = null;

    public static void main(String[] args) throws ConnectionException {
     CallWithHeader samples = new CallWithHeader();
        samples.createWithHeader();
    }

    public CallWithHeader() throws ConnectionException {
        metadataConnection = MetadataLoginUtil.login();
    }

    public void createWithHeader() throws ConnectionException {
        // Define two custom objects to be inserted.
        CustomObject co1 = new CustomObject();
        String name1 = "MyCustomObject1";
        co1.setFullName(name1 + "__c");
        co1.setDeploymentStatus(DeploymentStatus.Deployed);
        co1.setDescription("Created by the Metadata API");
        co1.setEnableActivities(true);
        co1.setLabel(name1 + " Object");
        co1.setPluralLabel(co1.getLabel() + "s");
        co1.setSharingModel(SharingModel.ReadWrite);

        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setLabel(co1.getFullName() + " Name");
        co1.setNameField(nf);
```

```
        // The second custom object doesn't have a Name field
        CustomObject co2 = new CustomObject();
        String name2 = "MyCustomObject2";
        co2.setFullName(name2 + "__c");
        co2.setDeploymentStatus(DeploymentStatus.Deployed);
        co2.setDescription("Created by the Metadata API");
        co2.setEnableActivities(true);
        co2.setLabel(name2 + " Object");
        co2.setPluralLabel(co2.getLabel() + "s");
        co2.setSharingModel(SharingModel.ReadWrite);


        // Setting the allOrNone header to true to cause
        //  the call to not commit any record if one or more
        //  records in this call have failures.
        metadataConnection.setAllOrNoneHeader(true);

        // Now that the header has been set, make the create call.
        SaveResult[] results = metadataConnection
                .createMetadata(new Metadata[] { co1, co2 });

        // Iterate through the call results
        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());
                }
            }
        }
    }

}
```

This is the output that the sample returns. The first record is rolled back and the second has a failure.

```
Errors were encountered while creating MyCustomObject1__c
Error message: Record rolled back because not all records were valid and the request was
using AllOrNone header
Status code: ALL_OR_NONE_OPERATION_ROLLED_BACK
Errors were encountered while creating MyCustomObject2__c
Error message: Must specify a nameField of type Text or AutoNumber
Status code: FIELD_INTEGRITY_EXCEPTION
```

CallOptions

Specifies the API client identifier.

Version

This call is available in all API versions.

Supported Calls

All Metadata API calls.

Fields

Field Name	Type	Description
`client`	string	A value that identifies an API client.

Sample Code—Java

To change the API client ID, add the `CallOptions` header to the metadata connection before you perform a call as follows:

```
metadataConnection.setCallOptions("client ID");
```

`DebuggingHeader`

Specifies that the deployment result will contain the debug log output, and specifies the level of detail included in the log. The debug log contains the output of Apex tests that are executed as part of a deployment.

Version

This header is available in all API versions.

Supported Calls

deploy()

Fields

Field Name	Type	Description
`categories`	LogInfo[]	A list of log categories with their associated log levels.
`debugLevel`	LogInfo (enumeration of type string)	This field has been deprecated and is only provided for backward compatibility. This field specifies the type of information returned in the debug log. The values are listed from the least amount of information returned to the most information returned. Valid values include: • `None`

Field Name	Type	Description
		• `Debugonly` • `Db` • `Profiling` • `Callout` • `Detail`

LogInfo

Field Name	Type	Description
`category`	LogCategory (enumeration of type string)	The category of operations for which to specify a debug log level. Valid values are: • `Db` • `Workflow` • `Validation` • `Callout` • `Apex_code` • `Apex_profiling` • `Visualforce` • `System` • `All`
`level`	LogCategoryLevel (enumeration of type string)	A log level that specifies how much information is logged for each category. Valid log levels are (listed from lowest to highest): • `NONE` • `ERROR` • `WARN` • `INFO` • `DEBUG` • `FINE` • `FINER` • `FINEST`

Sample Code—Java

Add the `DebuggingHeader` to the metadata connection before you perform the `deploy()` call as follows:

```
LogInfo[] logs = new LogInfo[1];
logs[0] = new LogInfo();
logs[0].setCategory(LogCategory.Apex_code);
logs[0].setLevel(LogCategoryLevel.Fine);
metadataConnection.setDebuggingHeader(logs, LogType.Detail);
```

The result of the deploy() call is obtained by calling checkDeployStatus(). After the deployment finishes, and if tests were run, the response of `checkDeployStatus()` contains the debug log output in an output header, `DebuggingInfo`, in the `debugLog` field of this header.

SessionHeader

Specifies the session ID that the login call returns. This session ID is used to authenticate all subsequent Metadata API calls.

Version

This header is available in all API versions.

Supported Calls

All Metadata API calls.

Fields

Field Name	Type	Description
`sessionId`	string	The session ID that the login call returns.

Sample Code—Java

Add the `SessionHeader` to the metadata connection before you perform a call as follows:

```
metadataConnection.setSessionHeader("<session_ID>");
```

APPENDICES

APPENDIX A CustomObjectTranslation Language Support: Fully Supported Languages

Not every language supports all the possible values for the fields in CustomObjectTranslation. Use this appendix to determine which field values a language supports.

Note: Salesforce offers three levels of language support: fully supported languages, end-user languages, and platform-only languages. This appendix provides information only for fully supported languages. For more information, see "Which Languages Does Salesforce Support?" in the Salesforce Help

Chinese (Simplified)

plural
false

caseType
Nominative

possessive
None

startwith
Consonant

plural
false

Chinese (Traditional)

caseType
Nominative

possessive
None

startwith
Consonant

plural
false

Danish

caseType
Nominative

article
Zero

Definite

Indefinite

possessive
None

gender
Feminine

Neuter

startwith
Consonant

plural
true

Dutch

CaseType
Nominative

article
Definite

Indefinite

gender
Feminine

Neuter

possessive
None

plural
true

Finnish

caseType
Ablative

Adessive

Allative

Dative

Elative

Essive

Genitive

Illative

Inessive

Nominative

Partitive

Translative

plural

true

possessive

None

First

Second

startwith

Consonant

French

article

Zero

Definite

Indefinite

gender

Masculine

Feminine

possessive

None

plural

true

startwith

Consonant

Vowel

German

article

Zero

Definite

Indefinite

caseType

Accusative

Dative

Genitive

Nominative

gender

Masculine

Feminine

Neuter

possessive

None

plural

true

Italian

article

Zero

Indefinite

Definite

CaseType

Nominative

gender

Masculine

Feminine

possessive

None

plural

true

startwith

Consonant

Vowel

Special

Japanese

CaseType

Nominative

possessive
None

startwith
Consonant

plural
false

Korean

CaseType
Nominative

possessive
None

startwith
Consonant

plural
false

Portuguese (Brazilian)

article
Zero

Definite

Indefinite

article
Zero

Indefinite

Definite

plural
true

Russian

caseType
Accusative

Dative

Genitive

Instrumental

Nominative

Prepositional

gender
Masculine
Feminine
Neuter
Animate_Masculine

plural
true
false

Spanish

article
Zero
Definite
Indefinite

CaseType
Nominative

gender
Masculine
Feminine

startwith
Consonant

plural
true

Thai

CaseType
Nominative

possessive
None

startwith
Consonant

plural
false

APPENDIX B CustomObjectTranslation Language Support: End-User Languages

Not every language supports all the possible values for the fields in CustomObjectTranslation. Use this appendix to determine which field values a language supports.

Note: Salesforce offers three levels of language support: fully supported languages, end-user languages, and platform-only languages. This appendix provides information only for end-user languages. For more information, see "Which Languages Does Salesforce Support?" in the Salesforce Help

Arabic

article
- Zero
- Definite

CaseType
- Nominative
- Accusative

gender
- Masculine
- Feminine

plural
- true

possessive
- None
- First
- Second

startwith
- Consonant

Bulgarian

article
- Zero
- Definite

CaseType
Nominative
Objective
gender
Masculine
Feminine
Neuter
possessive
None
plural
true
startwith
Consonant

Czech

CaseType
Accusative
Dative
Genitive
Instrumental
Locative
Vocative
Nominative
gender
Masculine
Feminine
Neuter
Animate_Masculine
plural
true

Greek

article
Zero
Definite
Indefinite

CaseType
Accusative
Genitive
Nominative
Vocative

gender
Masculine
Feminine
Neuter

possessive
None

plural
true

Hebrew

article
Zero
Definite

CaseType
Nominative

gender
Masculine
Feminine

possessive
None

plural
true

Hungarian

article
Zero
Definite
Indefinite

CaseType
Ablative
Accusative
Allative
Causalfinal

Dative

Delative

Distributive

Elative

Essiveformal

Illative

Inessive

Instrumental

Nominative

Sublative

Terminative

Translative

plural

true

possessive

None

First

Second

startwith

Consonant

Vowel

Indonesian

plural

true

CaseType

Nominative

Possessive

None

startwith

Consonant

Norwegian

article

Zero

Definite

Indefinite

CaseType
Nominative

gender
Masculine

Feminine

Neuter

possessive
None

plural
true

Polish

CaseType
Nominative

Accusative

Dative

Genitive

Instrumental

Locative

Vocative

gender
Masculine

Feminine

Neuter

Animate_Masculine

plural
true

Romanian

article
Zero

Definite

Indefinite

CaseType
Nominative

Dative

gender
Masculine
Feminine
Neuter

possessive
None

plural
true

Spanish (Mexico)

article
Zero
Definite
Indefinite

CaseType
Nominative

gender
Masculine
Feminine

possessive
None

plural
true

Turkish

article
Zero
Indefinite

CaseType
Ablative
Accusative
Dative
Genitive
Nominative

possessive
None
First
Second

startwith
Consonant

plural
true

Ukrainian

CaseType
Accusative

Dative

Genitive

Instrumental

Nominative

Locative

Vocative

gender
Masculine

Feminine

Neuter

Animate_Masculine

plural
true

Vietnamese

CaseType
Nominative

possessive
None

startwith
Consonant

plural
false

GLOSSARY

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

A

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

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

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

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

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

Application Programming Interface (API)
The interface that a computer system, library, or application provides to allow other computer programs to request services from it and exchange data.

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

B

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

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

C

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

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

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

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

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

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

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

Custom App
See App.

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

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

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

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

Custom S-Control

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

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

D

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

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

Data Manipulation Language (DML)
An Apex method or operation that inserts, updates, or deletes records.

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

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

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

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

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

E

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

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

Enterprise Edition
A Salesforce edition designed for larger, more complex businesses.

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

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

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

F

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

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

Filter Condition/Criteria
Condition on particular fields that qualifies items to be included in a list view or report, such as "State equals California."

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

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

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

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

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

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

G

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

H

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

I

ID
See Salesforce Record ID.

Inline S-Control

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

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

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

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

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

J

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

K

No Glossary items for this entry.

L

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

License Management Organization (LMO)
The Salesforce organization that you use to track all the Salesforce users who install your package. A license management organization must have the License Management Application (LMA) installed. It automatically receives notification every time your package is

installed or uninstalled so that you can easily notify users of upgrades. You can specify any Enterprise, Unlimited, Performance, or Developer Edition organization as your license management organization. For more information, go to http://www.salesforce.com/docs/en/lma/index.htm.

List View

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

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

Local Project

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

Locale

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

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

Logged-in User

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

Lookup Field

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

M

Managed Package

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

Manifest File

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

Manual Sharing

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

Many-to-Many Relationship

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

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

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

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

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

N

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

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

O

Object
An object allows you to store information in your Salesforce organization. The object is the overall definition of the type of information you are storing. For example, the case object allow you to store information regarding customer inquiries. For each object, your organization will have multiple records that store the information about specific instances of that type of data. For example, you might have a case record to store the information about Joe Smith's training inquiry and another case record to store the information about Mary Johnson's configuration issue.

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

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

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

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

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

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

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

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

P

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

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

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

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

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

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

Production Organization
A Salesforce organization that has live users accessing data.

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

Q

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

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

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

R

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

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

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

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

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

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

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

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

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

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

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

S

SaaS
See Software as a Service (SaaS).

S-Control

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

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

Salesforce Record ID
A unique 15- or 18-character alphanumeric string that identifies a single record in Salesforce.

Salesforce SOA (Service-Oriented Architecture)
A powerful capability of Force.com that allows you to make calls to external Web services from within Apex.

Sandbox
A nearly identical copy of a Salesforce production organization for development, testing, and training. The content and size of a sandbox varies depending on the type of sandbox and the editioin of the production organization associated with the sandbox.

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

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

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

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

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

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

- Apex-Managed Sharing—enables developers to programmatically manipulate sharing to support their application's behavior. See Apex-Managed Sharing.

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

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

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

Snippet

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

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

SOAP (Simple Object Access Protocol)
A protocol that defines a uniform way of passing XML-encoded data.

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

SOQL (Salesforce Object Query Language)
A query language that allows you to construct simple but powerful query strings and to specify the criteria that should be used to select data from the Force.com database.

SOSL (Salesforce Object Search Language)
A query language that allows you to perform text-based searches using the Force.com API.

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

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

T

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

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

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

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

U

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

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

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

URL (Uniform Resource Locator)
The global address of a website, document, or other resource on the Internet. For example, http://www.salesforce.com.

URL S-Control

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

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

V

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

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

W

Web Control
See URL S-Control.

Web Links
See Custom Links.

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

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

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

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

Workflow Action
A workflow action, such as an email alert, field update, outbound message, or task, fires when the conditions of a workflow rule are met.

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

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

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

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

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

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

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

X

XML (Extensible Markup Language)

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

Y

No Glossary items for this entry.

Z

Zip File

A data compression and archive format.

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

INDEX

D

E

F

G

H