Version 32.0: Winter '15

Force.com Tooling API Developer's Guide

Note: Any unreleased services or features referenced in this or other 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 upon features that are currently available.

Last updated: August 29, 2014

Table of Contents

GETTING STARTED **1**

Chapter 1: Introducing Tooling API **1**

When to Use Tooling API 1
Using Tooling REST API 2
Using Tooling SOAP API 5

REFERENCE **10**

Chapter 2: Object Reference **10**

Tooling API Objects 10
ApexClass 12
ApexClassMember 13
ApexCodeCoverage 16
ApexCodeCoverageAggregate 18
ApexComponent 19
ApexComponentMember 20
ApexExecutionOverlayAction 22
ApexExecutionOverlayResult 24
ApexLog 27
ApexOrgWideCoverage 29
ApexPage 30
ApexPageMember 30
ApexResult 32
ApexTestQueueItem 33
ApexTestResult 37
ApexTrigger 40
ApexTriggerMember 40
ContainerAsyncRequest 43
CustomField 45
CustomObject 47
DeployDetails 49
HistoryRetentionJob 49
FlexiPage 52
HeapDump 54
MenuItem 54
MetadataContainer 57
QuickActionDefinition 58
QuickActionList 62
QuickActionListItem 63
SOQLResult 65
StaticResource 66
SymbolTable 67

TraceFlag....70
ValidationRule....75
WorkflowRule....77

Index....79

GETTING STARTED

Chapter 1

Introducing Tooling API

Tooling API provides SOAP and REST interfaces that allow you to build custom development tools for Force.com applications.

For example, you can:

- Add features and functionality to your existing Force.com tools.
- Build dynamic modules for Force.com development into your enterprise integration tools.
- Build specialized development tools for a specific application or service.

Tooling API exposes objects used in developer tooling that you can access through REST or SOAP, and works just like the Salesforce REST API and SOAP API.

For detailed descriptions of Tooling API objects and the REST resources and SOAP calls that each object supports, see Tooling API Objects.

See Also:

When to Use Tooling API
Using Tooling REST API
Using Tooling SOAP API

When to Use Tooling API

Use Tooling API to perform the following tasks:

- Manage working copies of Apex classes and triggers and Visualforce pages and components using the ApexClassMember, ApexTriggerMember, ApexPageMember, ApexComponentMember and MetadataContainer objects.
- Manage working copies of static resource files using the StaticResource object.
- Check for updates and errors in working copies of Apex classes and triggers and Visualforce pages and components, and commit changes to your organization using the ContainerAsyncRequest object.
- Set heap dump markers using the ApexExecutionOverlayAction object.
- Overlay Apex code or SOQL statements on an Apex execution using the ApexExecutionOverlayAction object.
- Execute anonymous Apex. For sample code, see Using Tooling SOAP API and Using Tooling REST API.
- Set checkpoints to generate log files for yourself or for other users using the TraceFlag object.
- Access debug log and heap dump files using the ApexLog and ApexExecutionOverlayResult objects.
- Manage custom fields on custom objects using the CustomField object.
- Access code coverage results using the ApexCodeCoverage, ApexOrgWideCoverage and ApexCodeCoverageAggregate objects.

- Execute tests, and manage test results using the ApexTestQueueItem and ApexTestResult objects.
- Manage validation rules and workflow rules using the ValidationRule and WorkflowRule objects.

Selecting the Right API for Your Application

Tooling API provides both SOAP and REST interfaces.

- Use REST API if you're using a language that isn't strongly typed, like JavaScript. See Using Tooling REST API.
- Use SOAP API if you're using a strongly typed language like Java that generates Web service client code. See Using Tooling SOAP API.

See Also:

Using Tooling REST API
Using Tooling SOAP API

Using Tooling REST API

Use REST API if you're using a language that isn't strongly typed, like JavaScript. The REST Tooling API can be used just like the Force.com REST API; for details on usage, syntax, and authentication, see the *Force.com REST API Developer's Guide*.

Resources

This section lists supported REST resources in Tooling API.

The base URI for each Tooling REST API resource is `http://`***`domain`***`/services/data/`***`vXX.X`***`/tooling/` where ***`domain`*** is a Salesforce instance or a custom domain and ***`vXX.X`*** is the API version number. For example: `http://na1.salesforce.com/services/data/v28.0/tooling/`

Like the Force.com REST API, Tooling API uses the following resources.

URI	Supported Methods	Description
`/completions?type=`	GET	Retrieves available code completions of the referenced type. Currently only supports Apex system method symbols (`type=`**`apex`**). Available from API version 28.0 or later.
`/executeAnonymous/?anonymousBody=<url encoded body>`	GET	Executes Apex code anonymously. Available from API version 29.0 or later.
`/query/?q=`	GET	Executes a query against a Tooling API object and returns data that matches the specified criteria. If the query results are too large, the response contains the first batch of results and a query identifier. The identifier can be used in an additional request to retrieve the next batch.

`/runTestsAsynchronous/?classids=<comma separated list of class IDs>`
and
`/runTestsSynchronous/?classnames=<comma separated list of class names>`
	GET	Executes the tests in the specified classes. Running tests asynchronously allows methods to process in parallel, cutting down your test run times.

URI	Supported Methods	Description
/sobjects/	GET	Lists the available Tooling API objects and their metadata.
/sobjects/***SObjectName***/	GET POST	Describes the individual metadata for the specified object or creates a new record for a given object. For example, use the GET method to retrieve the metadata for the ApexExecutionOverlayAction object. Use the POST method to create a new ApexExecutionOverlayAction object.
/sobjects/***SObjectName***/describe/	GET	Completely describes the individual metadata at all levels for the specified object. For example, use this resource to retrieve the fields, URLs, and child relationships for a Tooling API object.
/sobjects/***SObjectName***/***id***/	GET PATCH DELETE	Accesses records based on the specified object ID. Use the GET method to retrieve records or fields, the DELETE method to delete records, and the PATCH method to update records.
/sobjects/**ApexLog/*id***/Body/	GET	Retrieves a raw debug log by ID. Available from API version 28.0 or later.

Examples

The following examples use Apex to execute REST requests, but you can use any standard REST tool to access Tooling REST API.

Note: Salesforce runs on multiple server instances. The examples in this guide use the *na1* instance. The instance your organization uses might be different.

First, set up the connection to your org and the HTTP request type:

```
HttpRequest req = new HttpRequest();
req.setHeader('Authorization', 'Bearer ' + UserInfo.getSessionID());
req.setHeader('Content-Type', 'application/json');
```

At the end of each request (examples below), add the following code to send the request and retrieve the body of the response:

```
Http h = new Http();
HttpResponse res = h.send(req);
system.debug(res.getBody());
```

To get a description of all available objects in Tooling API:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/sobjects/');
req.setMethod('GET');
```

To get a description of a specific Tooling API object, for example TraceFlag:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/sobjects/
TraceFlag/');
req.setMethod('GET');
```

To get a description of all the metadata for a specific Tooling API object, for example TraceFlag:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/sobjects/
TraceFlag/describe/');
req.setMethod('GET');
```

To create a new Tooling API object, for example MetadataContainer:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/sobjects/
MetadataContainer/');
req.setBody('{"Name":"TestContainer"}');
req.setMethod('POST');
```

Tip: Use the ID from this call in the rest of the examples.

To retrieve a Tooling API object by ID, for example MetadataContainer:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/sobjects/
MetadataContainer/ + containerID + '/');
req.setMethod('GET');
```

To update a Tooling API object by ID, for example MetadataContainer:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/sobjects/
MetadataContainer/ + containerID + '/');
req.setBody('{"Name":"NewlyNamedContainer"}');
req.setMethod('PATCH');
```

To query a Tooling API object by ID, for example MetadataContainer:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/query/?q=
Select+id,Name+from+MetadataContainer+Where+ID=\'' + containerID +  '\'');
req.setMethod('GET');
```

Or to query an object within a MetadataContainer:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/query/?q=
Select+id,Body,LastSyncDate,Metadata+from+ApexClassMember+Where+MetadataContainerID=\'
+ containerID +  '\'');
req.setMethod('GET');
```

To check on the status of a deployment, using ContainerAsyncRequest:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/sobjects/
ContainerAsyncRequest/' + requestID + '/');
req.setMethod('GET');
```

To execute anonymous Apex:

```
req.setEndpoint('http://na1.salesforce.com/services/data/v28.0/tooling/executeAnonymous/?
anonymousBody=System.debug('Test')%3B');
req.setMethod('GET');
```

See Also:

Tooling API Objects
Force.com REST API Developer's Guide

Using Tooling SOAP API

Use SOAP API if you're using a strongly typed language like Java that generates Web service client code. Tooling SOAP API is used just like the Salesforce SOAP API. For details on usage, syntax, and authentication, see the *SOAP API Developer's Guide*.

To access the Tooling API WSDL, from Setup, click **Develop** > **API** and click **Generate Tooling WSDL**.

Like the Salesforce SOAP API, Tooling API uses the following calls.

Call	Description
`create()`	Adds one or more new records to your organization's data.
`delete()`	Deletes one or more records from your organization's data.
`describeGlobal()`	Lists the available Tooling API objects and their metadata.
`describeSObjects()`	Describes metadata (field list and object properties) for the specified object or array of objects. Call `describeGlobal()` to retrieve a list of all Tooling API objects for your organization, then iterate through the list and use `describeSObjects()` to obtain metadata about individual objects.
`executeanonymous(string apexcode)`	Executes the specified block of Apex anonymously and returns the result.
`query()`	Executes a query against a Tooling API object and returns data that matches the specified criteria.
`retrieve()`	Retrieves one or more records based on the specified IDs.
`runTests()` and `runTestsAsynchronous()`	Executes test methods in the specified classes. Running tests asynchronously allows methods to process in parallel, cutting down your test run times. For example code, see ApexTestQueueItem.
`update()`	Updates one or more existing records in your organization's data.
`upsert()`	Creates new records and updates existing records; uses a custom field to determine the presence of existing records.

Examples

These examples use Java, but you can use Tooling SOAP API in any language that supports Web services.

Use `create()` to compile Apex classes or triggers in Developer Edition or sandbox organizations. The first example below uses ApexClass to compile a simple class with a single method called `SayHello`.

```
String classBody = "public class Messages {\n"
   + "public string SayHello() {\n"
   + " return 'Hello';\n" + "}\n"
   + "}";

// create a new ApexClass object and set the body
ApexClass apexClass = new ApexClass();
apexClass.Body = classBody;
ApexClass[] classes = { apexClass };

// call create() to add the class
SaveResult[] saveResults = sforce.create(classes);
for (int i = 0; i < saveResults.Length; i++)
   {
   if (saveResults[i].success)
      {
        Console.WriteLine("Successfully created Class: " +
         saveResults[i].id);
      }
   else
      {
         Console.WriteLine("Error: could not create Class ");
         Console.WriteLine("   The error reported was: " +
         saveResults[i].errors[0].message + "\n");
      }
   }
```

Use the `IsCheckOnly` parameter on ContainerAsyncRequest to indicate whether an asynchronous request should compile code without making any changes to the organization (true) or compile and save the code (false).

The example below expands upon the first by modifying the `SayHello()` method to accept a person's first and last name. This example uses MetadataContainer with ApexClassMember to retrieve and update the class, and ContainerAsyncRequest to compile and deploy the changes to the server. You can use the same method with ApexTriggerMember, ApexComponentMember, and ApexPageMember.

Note:

To test your code, modify the `IsCheckOnly` parameter in the code below, and log in to your org after a successful execution to verify the results.

- When `IsCheckOnly = true`, the `SayHello()` method should remain the same. (ApexClassMember contains the compiled results, but the class on the server remains the same.)
- When `IsCheckOnly = false`, the `SayHello()` method should show the change to accept a person's first and last name.

```
String updatedClassBody = "public class Messages {\n"
   + "public string SayHello(string fName, string lName) {\n"
   + " return 'Hello ' + fName + ' ' + lName;\n" + "}\n"
   + "}";

//create the metadata container object
MetadataContainer Container = new MetadataContainer();
Container.Name = "SampleContainer";

MetadataContainer[] Containers = { Container };
SaveResult[] containerResults = sforce.create(Containers);
if (containerResults[0].success)
{
   String containerId = containerResults[0].id;
```

```
      //create the ApexClassMember object
      ApexClassMember classMember = new ApexClassMember();
      //pass in the class ID from the first example
      classMember.ContentEntityId = classId;
      classMember.Body = updatedClassBody;
      //pass the ID of the container created in the first step
      classMember.MetadataContainerId = containerId;
      ApexClassMember[] classMembers = { classMember };

      SaveResult[] MembersResults = sforce.create(classMembers);
      if (MembersResults[0].success)
      {
         //create the ContainerAsyncRequest object
         ContainerAsyncRequest request = new ContainerAsyncRequest();
         //if the code compiled successfully, save the updated class to the server
         //change to IsCheckOnly = true to compile without saving
         request.IsCheckOnly = false;
         request.MetadataContainerId = containerId;
         ContainerAsyncRequest[] requests = { request };
         SaveResult[] RequestResults = sforce.create(requests);
         if (RequestResults[0].success)
         {
            string requestId = RequestResults[0].id;

            //poll the server until the process completes
            QueryResult queryResult = null;
            String soql = "SELECT Id, State, CompilerErrors, ErrorMsg FROM
ContainerAsyncRequest where id = '" + requestId + "'";
            queryResult = sforce.query(soql);
            if (queryResult.size > 0)
            {
               ContainerAsyncRequest _request =
(ContainerAsyncRequest)queryResult.records[0];
               while (_request.State.ToLower() == "queued")
               {
                  //pause the process for 2 seconds
                  Thread.Sleep(2000);

                  //poll the server again for completion
                  queryResult = sforce.query(soql);
                  _request = (ContainerAsyncRequest)queryResult.records[0];
               }

               //now process the result
               switch (_request.State)
               {
                  case "Invalidated":
                     break;

                  case "Completed":
                  //class compiled successfully
                  //see the next example on how to process the SymbolTable
                     break;

                  case "Failed":
               . .   break;

                  case "Error":
                     break;

                  case "Aborted":
                     break;

                  }
               }
               else
               {
                  //no rows returned
               }
```

```
            }
            else
            {
               Console.WriteLine("Error: could not create ContainerAsyncRequest object");
               Console.WriteLine("   The error reported was: " +
               RequestResults[0].errors[0].message + "\n");
            }
         }
         else
         {
            Console.WriteLine("Error: could not create Class Member ");
            Console.WriteLine("   The error reported was: " +
            MembersResults[0].errors[0].message + "\n");
         }
      }
      else
      {
      .. Console.WriteLine("Error: could not create MetadataContainer ");
         Console.WriteLine("   The error reported was: " +
         containerResults[0].errors[0].message + "\n");
      }
   }
```

Use a SymbolTable to access Apex class and trigger data in a structured format.

The example below queries the ApexClassMember object created in the previous example to obtain the SymbolTable of the modified class.

Note: The SOQL statement used depends on when the data is retrieved.

- To execute the query from within the example above, use the ID of the ContainerAsyncRequest. For example, `SELECT Body, ContentEntityId, SymbolTable FROM ApexClassMember where MetadataContainerId = '" + requestId + "'"`
- Otherwise, use the ID of the modified class as shown below. For example, `SELECT ContentEntityId, SymbolTable FROM ApexClassMember where ContentEntityId = '" + classId + "'"`

```
   //use the ID of the class from the previous step
   string classId = "01pA00000036itIIAQ";
   QueryResult queryResult = null;
   String soql = "SELECT ContentEntityId, SymbolTable FROM ApexClassMember where
ContentEntityId = '" + classId + "'";

   queryResult = sforce.query(soql);
   if (queryResult.size > 0)
   {
      ApexClassMember apexClass = (ApexClassMember)queryResult.records[0];
      SymbolTable symbolTable = apexClass.SymbolTable;

      foreach (Method _method in symbolTable.methods)
      {
         //here's the SayHello method
         String _methodName = _method.name;

         //is the method Global, Public or Private?
         String _methodVisibility = _method.visibility.ToString();

         //get the method's return type
         string _methodReturnType = _method.returnType;

         //get the fName & lName parameters
         foreach (Parameter _parameter in _method.parameters)
         {
            string _paramName = _parameter.name;
            string _parmType = _parameter.type;
         }
```

```
      }
   }
   else
   {
      //unable to locate class
   }
```

Use ApexExecutionOverlayAction to add checkpoints to your code for debugging.

This example adds a checkpoint to the class from the previous examples:

```
   //use the ID of the class from the first example.
   string classId = "01pA00000036itIIAQ";

   ApexExecutionOverlayAction action = new ApexExecutionOverlayAction();
   action.ExecutableEntityId = classId;
   action.Line = 3;
   action.LineSpecified = true;
   action.Iteration = 1;
   action.IterationSpecified = true;
   ApexExecutionOverlayAction[] actions = { action };

   SaveResult[] actionResults = sforce.create(actions);
   if (actionResults[0].success)
   {
      // checkpoint created successfully
   }
   else
   {
      Console.WriteLine("Error: could not create Checkpoint ");
      Console.WriteLine("   The error reported was: " +
      actionResults[0].errors[0].message + "\n");
   }
```

See Also:

Tooling API Objects

SOAP API Developer's Guide

REFERENCE

Chapter 2

Object Reference

This section provides a list of Tooling API objects, their fields, and supported SOAP API calls and REST resources. Not all fields are listed for all objects.

To verify the complete list of fields for an object, see the Tooling API WSDL. To access the Tooling API WSDL, from Setup, click **Develop** > **API** and click **Generate Tooling WSDL**.

Tooling API Objects

Tooling API includes the following objects:

Object	Description
ApexClass	Represents the saved copy of an Apex class. ApexClass uses the cached version of the class unless one is unavailable.
ApexClassMember	Represents the working copy of an Apex class for editing, saving or compiling in a MetadataContainer.
ApexCodeCoverage	Represents code coverage test results for an Apex class or trigger.
ApexCodeCoverageAggregate	Represents aggregate code coverage test results for an Apex class or trigger.
ApexComponent	Represents the saved copy of aVisualforce component. ApexComponent uses the cached version of the class unless one is unavailable.
ApexComponentMember	Represents the working copy of a Visualforce component for editing, saving or compiling in a MetadataContainer.
ApexExecutionOverlayAction	Specifies an Apex code snippet or SOQL query to execute at a specific line of code in an Apex class or trigger and optionally generate a heap dump.
ApexExecutionOverlayResult	Represents the result from the Apex code snippet or SOQL query defined in the associated ApexExecutionOverlayAction, and the resulting heap dump if one was returned.
ApexLog	Represents a debug log.
ApexPage	Represents the saved copy of an Apex page. ApexPage uses the cached version of the class unless one is unavailable.
ApexPageMember	Represents the working copy of a Visualforce page for editing, saving or compiling in a MetadataContainer.

Object	Description
ApexTestQueueItem	Represents a single Apex class in the Apex job queue.
ApexTestResult	Represents the result of an Apex test method execution.
ApexTrigger	Represents the saved copy of an Apex page. ApexTrigger uses the cached version of the class unless one is unavailable.
ApexTriggerMember	Represents the working copy of an Apex trigger for editing, saving or compiling in a MetadataContainer.
ContainerAsyncRequest	Allows you to compile and asynchronously deploy a MetadataContainer object to your organization.
CustomField	Represents a custom field on a custom object that stores data unique to your organization.
CustomObject	Represents a custom object that stores data unique to your organization. Includes access to the associated CustomObject object and related fields in Salesforce Metadata API.
HistoryRetentionJob	Represents the body of retained data from the archive, and the status of the archived data.
FlexiPage	Represents a Flexible Page. A Flexible Page is the home page for an app that appears as a menu item in the Salesforce1 navigation menu. Includes access to the associated FlexiPage object in the Salesforce Metadata API.
MenuItem	Represents a menu item.
MetadataContainer	Manages working copies of ApexClassMember, ApexTriggerMember, ApexPageMember and ApexComponentMember objects, including collections of objects that should be deployed together.
QuickActionDefinition	Represents the definition of a quick action.
QuickActionList	Represents a list of quick actions.
QuickActionListItem	Represents an item in a quick action list.
StaticResource	Represents the working copy of a static resource file for editing or saving. Static resources allow you to upload content that you can reference in a Visualforce page, including images, stylesheets, JavaScript, and other files.
TraceFlag	Represents a trace flag that triggers an Apex debug log at the specified logging level.
ValidationRule	Represents a formula that is used for specifying when a criteria is met. This includes both validation rules and workflow rules. Includes access to the associated ValidationRule object in the Salesforce Metadata API.
WorkflowRule	Represents a workflow rule that is used to fire off a specific workflow action when the specified criteria is met. Includes access to the associated WorkflowRule object in Salesforce Metadata API.

This section also provides details on the following complex types:

Object	Description
ApexResult	A complex type that represents the result of Apex code executed as part of an ApexExecutionOverlayAction, returned in an ApexExecutionOverlayResult.
DeployDetails	A complex type that contains detailed XML for any compile errors reported in the asynchronous request defined by a ContainerAsyncRequest object.
HeapDump	A complex type that represents a heap dump in an ApexExecutionOverlayResult.
SOQLResult	A complex type that represents the result of a SOQL query in an ApexExecutionOverlayResult.
SymbolTable	A complex type that represents all user-defined tokens in the `Body` of an ApexClass, ApexClassMember, or ApexTriggerMember and their associated line and column locations within the `Body`.

The following Tooling API objects are used internally by the Developer Console.

- IDEPerspective
- IDEWorkspace
- User.WorkspaceId

See Also:

Using Tooling REST API
Using Tooling SOAP API

ApexClass

Represents the saved copy of an Apex class. ApexClass uses the cached version of the class unless one is unavailable.

Available from API version 28.0 or later.

To edit, save, or compile Apex classes, use ApexClassMember.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`SymbolTable`	**Type** SymbolTable

Field Name	Details
	Properties Nillable **Description** A complex type that represents all user-defined tokens in the `Body` of an ApexClass, ApexClassMember, or ApexTriggerMember and their associated line and column locations within the `Body`. This field is null if the symbol table cannot be created.

Usage

To retrieve information about an Apex class, create an ApexClass object that references it. For example code, see Using Tooling SOAP API.

To edit, save, or compile Apex classes, use ApexClassMember.

Note: If there is not a cached version of SymbolTable, it will be compiled in the background and the query might take longer than expected. The SymbolTable returned from ApexClass does not contain references; to retrieve a SymbolTable with references, use ApexClassMember.

ApexClassMember

Represents the working copy of an Apex class for editing, saving or compiling in a MetadataContainer.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`FullName`	**Type** string **Properties** Group, Nillable **Description** The full name of the associated object in the Metadata API. Use to avoid race conditions on create, before you have IDs.
`Body`	**Type** string

Field Name	Details
	Properties Create, Update **Description** The data for the Apex class. The `Body` field is the only field you can `update()` or `PATCH`.
`Content`	**Type** string **Properties** None **Description** A string representation of ApexClassMetadata that lists the version, status, and packaged versions of the corresponding Apex class.
`ContentEntityId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** A reference to an Apex class. There can be only one `ContentEntityId` per ApexClassMember, otherwise, an error is reported. This field is required.
`LastSyncDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date and time that this ApexClassMember `Body` was replicated from the underlying Apex class. When you deploy a MetadataContainer, this value is compared with the `LastModifiedDate` of the underlying Apex class. If `LastSyncDate` is older than `LastModifiedDate`, the deployment fails with an error.
`Metadata`	**Type** ApexClassMetadata

Field Name	Details
	Properties None **Description** An object that describes the version, status, and packaged versions of the corresponding Apex class.
`MetadataContainerId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** A reference to a MetadataContainer or ContainerAsyncRequest object. As part of a successful deployment, this field is reset from the ID of the deployed MetadataContainer to the ID of the corresponding ContainerAsyncRequest object. This field is required.
`SymbolTable`	**Type** SymbolTable **Properties** Nillable **Description** A complex type that represents all user-defined tokens in the `Body` of an ApexClass, ApexClassMember, or ApexTriggerMember and their associated line and column locations within the `Body`. This field is null if the symbol table cannot be created. A symbol table cannot be created if the content referenced by the `ContentEntityId` field doesn't use a symbol table, or if compiler errors for the last deployment of the MetadataContainer in the `MetadataContainerId` field prevented a symbol table from being created.

Usage

To edit, save, or compile an Apex class, create an ApexClassMember object that references it.

Note: Once an ApexClassMember is successfully deployed in a MetadataContainer, the `MetadataContainerId` is changed to the ID of the ContainerAsyncRequest, and the ApexClassMember can't be modified or reused.

Apex classes are often dependent on each other for functionality. For example, a method in one class can call a method in another class. If source file A is dependent on modified source file B and you try to save and compile source file A before you've saved the changes to source file B, the compiler will throw an error. To successfully save and compile a group of related source files, put the corresponding ApexClassMember and ApexTriggerMember objects in a single MetadataContainer object.

Each ApexClassMember object can only refer to a single MetadataContainer object. Multiple ApexClassMember objects can refer to the same MetadataContainer object.

ApexCodeCoverage

Represents code coverage test results for an Apex class or trigger.

Available in Tooling API version 29.0 and later.

Supported SOAP API Calls

`describeSObjects()`, `query()`, `retrieve()`

Supported REST API HTTP Methods

`Query`, `GET`

Fields

Field	Details
`ApexTestClassId`	**Type** string **Properties** Filter, Group, Sort **Description** The ID of the test class.
`TestMethodName`	**Type** string **Properties** Filter, Group, Sort **Description** The name of the test method.
`ApexClassorTriggerId`	**Type** string **Properties** Filter, Group, Sort **Description** The ID of the class or trigger under test.
`NumLinesCovered`	**Type** int

Field	Details
	Properties Filter, Group, Sort **Description** The number of covered lines.
`NumLinesUncovered`	**Type** int **Properties** Filter, Group, Sort **Description** The number of uncovered lines.
`Coverage`	**Type** complexvalue **Properties** None **Description** Two lists of integers. The first is the covered lines, and the second is the list of uncovered lines. If a lines is missing from both lists, the line is not executable and does not require coverage. Coverage includes the following fields: • `coveredLines` • `namespace` • `uncoveredLines`

Usage

To query for code coverage, specify an Apex class, test class, or both. The returned JSON or XML object will contain two lists of integers: one for covered and one for uncovered lines.

The following example SOQL query retrieves code coverage results for a specific class or trigger covered by a specific test class:

```
SELECT Coverage
FROM ApexCodeCoverage
WHERE ApexClassOrTrigger = '01pD000000066GR'
AND ApexTestClass = '01pD000000064pu'
```

For per-class code coverage, the query would be:

```
SELECT Coverage
FROM ApexCodeCoverage
WHERE ApexClassOrTrigger = '01pD000000066GR'
```

Note: In this case, multiple rows may be returned, since there may be multiple test classes that cover the same test class.

As noted above, `Coverage` is returned as two lists of integers. The first is the covered lines, and the second is the list of uncovered lines. If a line is missing from both lists, the line is not executable and does not require coverage. For example, if the covered lines are 2, 9, and 11, and uncovered lines are 3, 4, 5, and 6; the result would be: `{2,9,11},{3,4,5,6}`. The missing lines (1, 7, 8 and 10) are not executable.

Code coverage percentage is a simple calculation of the number of covered lines divided by the sum of the number of covered lines and the number of uncovered lines. For example, to calculate code coverage percentage in SOAP:

```
ApexCodeCoverage acc = null; //Query for an ApexCodeCoverage object
Coverage coverage = acc.coverage;
int[] covered = coverage.coveredLines;
int[] uncovered = coverage.uncoveredLines;
int percent = covered.length / (covered.length + uncovered.length);
System.out.println("Total class coverage is " + percent + "%.");
```

ApexCodeCoverageAggregate

Represents aggregate code coverage test results for an Apex class or trigger.

Available in Tooling API version 29.0 and later.

Supported SOAP API Calls

`describeSObjects()`, `query()`, `retrieve()`

Supported REST API HTTP Methods

`Query`, `GET`, `DELETE`

Fields

Field	Details
`ApexClassorTriggerId`	**Type** string **Properties** Filter, Group, Sort **Description** The ID of the class or trigger under test.
`NumLinesCovered`	**Type** int **Properties** Filter, Group, Sort **Description** The number of covered lines.

Field	Details
NumLinesUncovered	**Type** int **Properties** Filter, Group, Sort **Description** The number of uncovered lines.
Coverage	**Type** complexvalue **Properties** None **Description** Two lists of integers. The first is the covered lines, and the second is the list of uncovered lines. If a lines is missing from both lists, the line is not executable and does not require coverage. Coverage includes the following fields: • coveredLines • namespace • uncoveredLines

Usage

To query for aggregate code coverage, specify an Apex test class. The returned JSON or XML object will contain two lists of integers: one for covered and one for uncovered lines. For examples, see ApexCodeCoverage.

ApexComponent

Represents the saved copy of aVisualforce component. ApexComponent uses the cached version of the class unless one is unavailable.

Available from API version 28.0 or later.

To edit, save, or compile Visualforce components, use ApexComponentMember.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Usage

To retrieve information about a Visualforce component, create an ApexComponent object that references it. For example code, see Using Tooling SOAP API.

To edit, save, or compile Visualforce components, use ApexComponentMember.

ApexComponentMember

Represents the working copy of a Visualforce component for editing, saving or compiling in a MetadataContainer.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`FullName`	**Type** string **Properties** Group, Nillable **Description** The full name of the associated object in the Metadata API. Use to avoid race conditions on create, before you have IDs.
`Body`	**Type** string **Properties** Create, Update **Description** The data for the Visualforce component. The `Body` field is the only field you can `update()` or `PATCH`.
`Content`	**Type** string **Properties** None **Description** A string representation of ApexComponentMetadata that lists the version, status, and packaged versions of the corresponding Visualforce component.
`ContentEntityId`	**Type** reference

Field Name	Details
	Properties Create, Filter, Group, Sort **Description** A reference to a Visualforce component. There can be only one `ContentEntityId` per ApexComponentMember, otherwise, an error is reported. This field is required.
`LastSyncDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date that this ApexComponentMember `Body` was replicated from the underlying entity. When you deploy a MetadataContainer, this value is compared with the `LastModifiedDate` of the underlying Visualforce component. If `LastSyncDate` is older than `LastModifiedDate`, the deployment fails with an error.
`Metadata`	**Type** ApexComponentMetadata **Properties** None **Description** An object that describes the version, status, and packaged versions of the corresponding Visualforce component.
`MetadataContainerId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** A reference to a MetadataContainer or ContainerAsyncRequest object. As part of a successful deployment, this field is reset from the ID of the deployed MetadataContainer to the ID of the corresponding ContainerAsyncRequest object. This field is required.

Usage

To edit, save, or compile a Visualforce component, create an ApexComponentMember object that references it. To create a new Visualforce component, use the Force.com REST API or the Metadata API.

Note: Once an ApexComponentMember is successfully deployed in a MetadataContainer, the `MetadataContainerId` is changed to the ID of the ContainerAsyncRequest, and the ApexComponentMember can't be modified or reused.

Visualforce pages and components are often dependent on each other for functionality. To successfully save and compile a group of related source files, put the corresponding ApexComponentMember and ApexPageMember objects in a single MetadataContainer object.

Each ApexComponentMember object can only refer to a single MetadataContainer object. Multiple ApexComponentMember objects can refer to the same MetadataContainer object.

See Also:

Force.com REST API Developer's Guide
Metadata API Developer's Guide

ApexExecutionOverlayAction

Specifies an Apex code snippet or SOQL query to execute at a specific line of code in an Apex class or trigger and optionally generate a heap dump.

Supported SOAP Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`ActionScript`	**Type** string **Properties** Create, Nillable, Update **Description** The Apex code or SOQL query to run when execution reaches the line in the Apex class or trigger at the specified iteration. Results will be included in the heap dump file.
`ActionScriptType`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update

Field Name	Details
	Description Indicates whether the `ActionScript` is written in Apex or SOQL. Valid values are: • `None` • `Apex` • `SOQL` This field is required.
`ExecutableEntityId`	**Type** reference **Properties** Create, Filter, Group, Sort, **Description** The ID of the Apex class or trigger being executed. This field is required.
`ExpirationDate`	**Type** dateTime **Properties** Create, Filter, Sort, Update **Description** The expiration date of the overlay action. This field is required.
`IsDumpingHeap`	**Type** boolean **Properties** Create, Defaulted on create, Filter, Group, Sort, Update **Description** Indicates whether a heap dump is generated (`true`) or not (`false`). To execute the `ActionScript` without generating a heap dump, set this field to `false`. This field is required.
`Iteration`	**Type** int **Properties** Create, Filter, Group, Sort, Update **Description** The number of times the specified line should execute before the heap dump is generated. This field is required.

Field Name	Details
`Line`	**Type** int **Properties** Create, Filter, Group, Sort, Update **Description** The line number of the heap dump marker. This field is required.
`ScopeId`	**Type** reference **Properties** Create, Filter, Group, Sort, Update **Description** The user who executed the action. This field is required.

Usage

When you are troubleshooting a runtime issue, you often want to find out more about the state of a variable or the state of the database, or create a specific condition to test your code. Use ApexExecutionOverlayAction to overlay a diagnostic output on an Apex class or trigger without compromising production code.

ApexExecutionOverlayResult

Represents the result from the Apex code snippet or SOQL query defined in the associated ApexExecutionOverlayAction, and the resulting heap dump if one was returned.

Available from API version 28.0 or later.

Supported SOAP Calls

`query()`, `retrieve()`, `delete()`

Supported REST HTTP Methods

`Query`, `GET`, `DELETE`

Fields

Field Name	Details
`ActionScript`	**Type** string **Properties** Nillable

Field Name	Details
	Description The Apex code or SOQL query that was run.
`ActionScriptType`	**Type** picklist **Properties** Filter, Group, Sort, Nillable **Description** Indicates whether the `ActionScript` is written in Apex or SOQL. Valid values are: • `None` • `Apex` • `SOQL`
`ApexResult`	**Type** ApexResult **Properties** Nillable **Description** A complex type that represents the result of Apex code executed as part of an ApexExecutionOverlayAction, returned in an ApexExecutionOverlayResult.
`ExpirationDate`	**Type** dateTime **Properties** Filter, Sort **Description** The expiration date of the overlay action.
`HeapDump`	**Type** HeapDump **Properties** Nillable **Description** A complex type that represents a heap dump in an ApexExecutionOverlayResult.
`IsDumpingHeap`	**Type** boolean

Field Name	Details
	Properties Defaulted on create, Filter, Group, Sort **Description** Indicates whether a heap dump was generated (`true`) or not (`false`).
`Iteration`	**Type** int **Properties** Create, Filter, Group, Sort, Update **Description** The number of times the specified line should execute before the heap dump is generated. This field is required.
`Line`	**Type** int **Properties** Filter, Group, Sort, Nillable **Description** The line number of the checkpoint.
`SOQLResult`	**Type** SOQLResult **Properties** Nillable **Description** A complex type that represents the result of a SOQL query in an ApexExecutionOverlayResult.
`UserId`	**Type** reference **Properties** Filter, Group, Sort, **Description** The user who executed the action.

Usage

When you are troubleshooting a runtime issue, you often want to find out more about the state of a variable or the state of the database, or create a specific condition to test your code. Use ApexExecutionOverlayAction to overlay a diagnostic output

on an Apex class or trigger without compromising production code, and use ApexExecutionOverlayResult to navigate the results.

ApexLog

Represents a debug log.

To retrieve a raw log by ID, use the REST resource: `/sobjects/ApexLog/id/Body/`. (Available from API version 28.0 or later.)

Supported SOAP Calls

`delete()`, `describeSObjects()`, `query()`, `retrieve()`

Supported REST HTTP Methods

`Query`, `GET`, `DELETE`

Fields

Field	Details
`Application`	**Type** textarea **Properties** Filter, Group, Sort **Description** This value depends on the client type that triggered the log or heap dump. • For API clients, this value is the client ID. • For browser clients, this value is `Browser`. This field is required.
`DurationMilliseconds`	**Type** int **Properties** Filter, Group, Sort **Description** The duration of the transaction in milliseconds. This field is required.
`Location`	**Type** picklist **Properties** Filter, Group, Sort, Nillable, Restricted picklist **Description** Specifies the location of the origin of the log or heap dump. Values are:

Field	Details
	• `Monitoring` — Generated as part of debug log monitoring and visible to all administrators. These types of logs are maintained until the user or the system overwrites them. • `SystemLog` — Generated as part of system log monitoring and visible only to you. These types of logs are only maintained for 60 minutes or until the user clears them. • `Preserved` — A system log that is maintained longer than 60 minutes. Used for internal support.
`LogLength`	**Type** int **Properties** Filter, Group, Sort **Description** Length of the log or heap dump in bytes. This field is required.
`LogUserId`	**Type** reference **Properties** Filter, Group, Sort, Nillable **Description** ID of the user whose actions triggered the debug log or heap dump.
`Operation`	**Type** string **Properties** Filter, Group, Sort **Description** Name of the operation that triggered the debug log or heap dump, such as `APEXSOAP`, `Apex Sharing Recalculation`, and so on. This field is required.
`Request`	**Type** string **Properties** Filter, Group, Sort **Description** Request type. Values are: • `API` — Request came from an API. • `Application` — Request came from the Salesforce user interface. This field is required.

Field	Details
`StartTime`	**Type** dateTime **Properties** Filter, Sort **Description** Start time of the transaction. This field is required.
`Status`	**Type** string **Properties** Filter, Group, Sort **Description** Status of the transaction. This value is either `Success`, or the text of an unhandled Apex exception. This field is required.

ApexOrgWideCoverage

Represents code coverage test results for an entire organization.

Available in Tooling API version 29.0 and later.

Supported SOAP API Calls

`describeSObjects()`, `delete()`, `query()`, `retrieve()`

Supported REST API HTTP Methods

`Query`, `GET`, `DELETE`

Fields

Field	Details
`PercentCovered`	**Type** int **Properties** Filter, Group, Nillable, Sort **Description** The percentage of the code in the organization that is covered by tests.

ApexPage

Represents the saved copy of an Apex page. ApexPage uses the cached version of the class unless one is unavailable.

Available from API version 28.0 or later.

To edit, save, or compile Apex pages, use ApexPageMember.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Usage

To retrieve information about an Apex page, create an ApexPage object that references it. For example code, see Using Tooling SOAP API.

To edit, save, or compile Apex pages, use ApexPageMember.

ApexPageMember

Represents the working copy of a Visualforce page for editing, saving or compiling in a MetadataContainer.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`FullName`	**Type** string **Properties** Group, Nillable **Description** The full name of the associated object in the Metadata API. Use to avoid race conditions on create, before you have IDs.
`Body`	**Type** string **Properties** Create, Update

Field Name	Details
	Description The data for the Visualforce page. The `Body` field is the only field you can `update()` or `PATCH`.
`Content`	**Type** string **Properties** None **Description** A string representation of ApexPageMetadata that lists the version, status, and packaged versions of the corresponding Visualforce page.
`ContentEntityId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** A reference to a Visualforce page. There can be only one `ContentEntityId` per ApexPageMember, otherwise, an error is reported. This field is required.
`LastSyncDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date that this ApexPageMember `Body` was replicated from the underlying entity. When you deploy a MetadataContainer, this value is compared with the `LastModifiedDate` of the underlying Visualforce page. If `LastSyncDate` is older than `LastModifiedDate`, the deployment fails with an error.
`Metadata`	**Type** ApexPageMetadata **Properties** None

Field Name	Details
	Description An object that describes the version, status, and packaged versions of the corresponding Visualforce page.
`MetadataContainerId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** A reference to a MetadataContainer or ContainerAsyncRequest object. As part of a successful deployment, this field is reset from the ID of the deployed MetadataContainer to the ID of the corresponding ContainerAsyncRequest object. This field is required.

Usage

To edit, save, or compile a Visualforce page, create an ApexPageMember object that references it. To create a new Visualforce page, use the Force.com REST API or the Metadata API.

Note: Once an ApexPageMember is successfully deployed in a MetadataContainer, the `MetadataContainerId` is changed to the ID of the ContainerAsyncRequest, and the ApexPageMember can't be modified or reused.

Visualforce pages and components are often dependent on each other for functionality. To successfully save and compile a group of related source files, put the corresponding ApexPageMember and ApexComponentMember objects in a single MetadataContainer object. Use ContainerAsyncRequest to send the MetadataContainer to the application server.

Each ApexPageMember object can only refer to a single MetadataContainer object. Multiple ApexPageMember objects can refer to the same MetadataContainer object.

See Also:

Force.com REST API Developer's Guide
Metadata API Developer's Guide

ApexResult

A complex type that represents the result of Apex code executed as part of an ApexExecutionOverlayAction, returned in an ApexExecutionOverlayResult.

Available from API version 28.0 or later.

Fields

Field	Details
`apexError`	**Type** string **Description** The error text returned if the execution was unsuccessful.
`apexExecutionResult`	**Type** ExecuteAnonymousResult **Description** The structured result returned from a successful execution. ExecuteAnonymousResult includes the following fields: • `column` • `compileProblem` • `compiled` • `exceptionMessage` • `exceptionStackTrace` • `line` • `success` **Note:** ExecuteAnonymousResult is outside the current execution context; it does not provide access to variables in the heap.

Usage

Overlay Apex on checkpoints to capture structured debugging information.

ApexTestQueueItem

Represents a single Apex class in the Apex job queue.

Available from API version 30.0 or later.

Supported SOAP API Calls

`create()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`

Fields

Field Name	Details
`ApexClassId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** The Apex class whose tests are to be executed. This field can't be updated.
`Status`	**Type** picklist **Properties** Filter, Group, Restricted picklist, Sort, Update **Description** The status of the test. Valid values are: • `Queued` • `Processing` • `Aborted` • `Completed` • `Failed` • `Preparing` • `Holding` To abort a class that is in the Apex job queue, perform an update operation on the `ApexTestQueueItem` object and set its `Status` field to `Aborted`.
`ExtendedStatus`	**Type** string **Properties** Filter, Sort, Nillable **Description** The pass rate of the test run. For example: “(4/6)”. This means that four out of a total of six tests passed. If the class fails to execute, this field contains the cause of the failure.
`ParentJobId`	**Type** reference

Field Name	Details
	Properties Filter, Group, Sort, Nillable, **Description** Read-only. Points to the `AsyncApexJob` that represents the entire test run. If you insert multiple Apex test queue items in a single bulk operation, the queue items will share the same parent job. This means that a test run can consist of the execution of the tests of several classes if all the test queue items are inserted in the same bulk operation.

Usage

Insert an ApexTestQueueItem object to place its corresponding Apex class in the Apex job queue for execution. The Apex job executes the test methods in the class.

The example `RunTestListener.java` class below subscribes to the TestResult system topic and prints out the test results using ApexTestQueueItem and ApexTestResult. The example assumes the following:

- You have already set up a Java client application for Streaming API. This example uses the `org.cometd.client.BayeuxClient` created in the Java Client code example in the *Streaming API Developer's Guide*.
- You have a logged in `com.sforce.soap.tooling.SoapConnection`. For examples, see the *SOAP API Developer's Guide*.

Note:

The `RunTestListener.java` class must be instantiated after the Streaming API handshake. For example:

```
SoapConnection toolingConn = //Already set and logged in;
BayeuxClient client = //Already set and logged in;

//Listen on the handshake event
boolean handshaken = client.waitFor(10 * 1000, BayeuxClient.State.CONNECTED);
if (!handshaken) {
   System.out.println("Failed to handshake: " + client);
   System.exit(1);
}
final RunTestListener = null;
client.getChannel(Channel.META_SUBSCRIBE).addListener(
   new ClientSessionChannel.MessageListener() {
      public void onMessage(ClientSessionChannel channel, Message message) {
         boolean success = message.isSuccessful();
         if (success) {
            //Replace with your own ApexClass ids
            String apexTestClassId1 = "01pD00000007M0CIAU";
            String apexTestClassId2 = "01pD00000007NqtIAE";
            listener.runTests(new String[]{apexTestClassId1, apexTestClassId2});
         }
      }
   };
);
//This will subscribe to the TestRun system topic
listener = new RunTestListener(client, toolingConn);
```

```
import java.util.HashMap;
import org.cometd.bayeux.Message;
```

```
import org.cometd.bayeux.client.ClientSessionChannel;
import org.cometd.bayeux.client.ClientSessionChannel.MessageListener;
import org.cometd.client.BayeuxClient;

import com.sforce.soap.tooling.ApexTestQueueItem;
import com.sforce.soap.tooling.ApexTestResult;
import com.sforce.soap.tooling.QueryResult;
import com.sforce.soap.tooling.SObject;
import com.sforce.soap.tooling.SoapConnection;
import com.sforce.ws.ConnectionException;

public class RunTestListener {
   private static final String CHANNEL = "/systemTopic/TestResult";
   private SoapConnection conn;

   public RunTestListener(BayeuxClient client, SoapConnection conn) {
      this.conn = conn;
      System.out.println("Subscribing for channel: " + CHANNEL);
      client.getChannel(CHANNEL).subscribe(new MessageListener() {
         @Override
         public void onMessage(ClientSessionChannel channel, Message message) {
            HashMap data = (HashMap) message.getData();
            HashMap sobject = (HashMap) data.get("sobject");
            String id = (String) sobject.get("Id");
            System.out.println("\nAysncApexJob " + id);
            getTestQueueItems(id);
         }
     });
   }

   public void runTests(String[] apexTestClassIds) {
      if (apexTestClassIds.length == 0) {
         System.out.println("No test to run");
         return;
      }
      System.out.println("Running async test run");
      String ids = apexTestClassIds[0];
      for (int i = 1; i < apexTestClassIds.length; i++) {
         ids += ","+apexTestClassIds[i];
      }
      try {
         conn.runTestsAsynchronous(ids);
      } catch (ConnectionException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
   }
   private void getTestQueueItems(String asyncApexJobId) {
      try {
         QueryResult res = conn
           .query("SELECT Id, Status, ApexClassId FROM ApexTestQueueItem WHERE ParentJobId
 = '"
               + asyncApexJobId + "'");
         if (res.getSize() > 0) {
            for (SObject o : res.getRecords()) {
               ApexTestQueueItem atqi = (ApexTestQueueItem) o;
               System.out.println("\tApexTestQueueItem - "+atqi.getStatus());
               if (atqi.getStatus().equals("Completed")) {
                  getApexTestResults(atqi.getId());
               }
            }
         } else {
            System.out.println("No queued items for " + asyncApexJobId);
         }
      } catch (ConnectionException e) {
         e.printStackTrace();
      }
   }
```

```
   private void getApexTestResults(String apexTestQueueItemId) {
      try {
         QueryResult res = conn
           .query("SELECT StackTrace,Message, AsyncApexJobId,MethodName, Outcome,ApexClassId
 FROM ApexTestResult WHERE QueueItemId = '"
               + apexTestQueueItemId + "'");
         if (res.getSize() > 0) {
            for (SObject o : res.getRecords()) {
               ApexTestResult atr = (ApexTestResult) o;
               System.out.println("\tTest result for "
                  + atr.getApexClassId() + "." + atr.getMethodName());
               String msg = atr.getOutcome().equals("Fail") ? " - "
                  + atr.getMessage() + " " + atr.getStackTrace() : "";
               System.out.println("\t\tTest " + atr.getOutcome() + msg);
            }
         } else {
            System.out.println("No Test Results for " + apexTestQueueItemId);
         }
      } catch (ConnectionException e) {
         // TODO Auto-generated catch block
          e.printStackTrace();
      }
   }
}
```

ApexTestResult

Represents the result of an Apex test method execution.

Available from API version 30.0 or later.

Supported SOAP API Calls

`describeSObjects()`, `query()`, `retrieve()`

Supported REST API HTTP Methods

`Query`, `GET`

Fields

Field Name	Details
`ApexClassId`	**Type** reference **Properties** Filter, Group, Sort **Description** The Apex class whose test methods were executed.
`ApexLogId`	**Type** reference

Field Name	Details
	Properties Filter, Group, Nillable, Sort **Description** Points to the `ApexLog` for this test method execution if debug logging is enabled; otherwise, `null`.
`AsyncApexJobId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** Read-only. Points to the `AsyncApexJob` that represents the entire test run. This field points to the same object as `ApexTestQueueItem.ParentJobId`.
`Message`	**Type** string **Properties** Filter, Nillable, Sort **Description** The exception error message if a test failure occurs; otherwise, `null`.
`MethodName`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The name of the test method.
`Outcome`	**Type** picklist **Properties** Filter, Group, Restricted picklist, Sort

Field Name	Details
	Description The result of the test. Valid values are: • `Pass` • `Failed` • `CompileFail` • `Skip`
`QueueItemId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** Points to the ApexTestQueueItem which is the class that this test method is part of.
`StackTrace`	**Type** string **Properties** Filter, Nillable, Sort **Description** The Apex stack trace if the test failed; otherwise, `null`.
`TestTimestamp`	**Type** dateTime **Properties** Filter, Sort **Description** The start time of the test method.

Usage

You can query the fields of the ApexTestResult record that corresponds to a test method executed as part of an Apex class execution.

Each test method execution is represented by a single ApexTestResult record. For example, if an Apex test class contains six test methods, six ApexTestResult records are created. These records are in addition to the ApexTestQueueItem record that represents the Apex class.

For example code, see ApexTestQueueItem.

ApexTrigger

Represents the saved copy of an Apex page. ApexTrigger uses the cached version of the class unless one is unavailable.

Available from API version 28.0 or later.

To edit, save, or compile Apex triggers, use ApexTriggerMember.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Usage

To retrieve information about an Apex trigger, create an ApexTrigger object that references it. For example code, see Using Tooling SOAP API.

To edit, save, or compile Apex triggers, use ApexTriggerMember.

ApexTriggerMember

Represents the working copy of an Apex trigger for editing, saving or compiling in a MetadataContainer.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`FullName`	**Type** string **Properties** Group, Nillable **Description** The full name of the associated object in the Metadata API. Use to avoid race conditions on create, before you have IDs.
`Body`	**Type** string **Properties** Create, Update

Field Name	Details
	Description The data for the Apex trigger. The `Body` field is the only field you can `update()` or `PATCH`.
`Content`	**Type** string **Properties** None **Description** A string representation of ApexTriggerMetadata that lists the version, status, and packaged versions of the corresponding Apex trigger.
`ContentEntityId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** A reference to an Apex trigger. There can be only one `ContentEntityId` per ApexTriggerMember, otherwise, an error is reported. This field is required.
`LastSyncDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date that this ApexTriggerMember `Body` was replicated from the underlying entity. When you deploy a MetadataContainer, this value is compared with the `LastModifiedDate` of the underlying Apex trigger. If `LastSyncDate` is older than `LastModifiedDate`, the deployment fails with an error.
`Metadata`	**Type** ApexTriggerMetadata **Properties** None

Field Name	Details
	Description An object that describes the version, status, and packaged versions of the corresponding Apex trigger.
`MetadataContainerId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** A reference to a MetadataContainer or ContainerAsyncRequest object. As part of a successful deployment, this field is reset from the ID of the deployed MetadataContainer to the ID of the corresponding ContainerAsyncRequest object. This field is required.
`SymbolTable`	**Type** SymbolTable **Properties** Nillable **Description** A complex type that represents all user-defined tokens in the `Body` of an ApexClass, ApexClassMember, or ApexTriggerMember and their associated line and column locations within the `Body`. This field is null if the symbol table cannot be created. A symbol table cannot be created if the content referenced by the `ContentEntityId` field doesn't use a symbol table, or if compiler errors for the last deployment of the MetadataContainer in the `MetadataContainerId` field prevented a symbol table from being created.

Usage

To edit, save, or compile an Apex trigger, create an ApexTriggerMember object that references it. To create a new trigger, use the Force.com REST API or the Metadata API.

Note: Once an ApexTriggerMember is successfully deployed in a MetadataContainer, the `MetadataContainerId` is changed to the ID of the ContainerAsyncRequest, and the ApexTriggerMember can't be modified or reused.

Apex triggers and classes are often dependent on each other for functionality. For example, a method in one class can call a method in another class. If source file A is dependent on modified source file B and you try to save and compile source file A before you've saved the changes to source file B, the compiler will throw an error. To successfully save and compile a group of related source files, put the corresponding ApexTriggerMember and ApexClassMember objects in a single MetadataContainer object. Use ContainerAsyncRequest to send the MetadataContainer to the application server.

Each ApexTriggerMember object can only refer to a single MetadataContainer object. Multiple ApexTriggerMember objects can refer to the same MetadataContainer object.

See Also:

Force.com REST API Developer's Guide
Metadata API Developer's Guide

ContainerAsyncRequest

Allows you to compile and asynchronously deploy a MetadataContainer object to your organization.

Supported SOAP API Calls

`create()`, `describeSObjects()`, `query()`, `retrieve()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`

Fields

Field Name	Details
`DeployDetails`	**Type** DeployDetails **Properties** Nillable **Description** Provides detailed XML for any compile errors reported during an asynchronous request. Includes `componentFailures`. Replaces the JSON field `CompilerErrors` in Tooling API version 31.0 and later.
`ErrorMsg`	**Type** textarea **Properties** Nillable **Description** Errors reported during an asynchronous request.
`IsCheckOnly`	**Type** boolean **Properties** Create, Defaulted on create, Filter, Group, Sort

Field Name	Details
	Description Indicates whether the asynchronous request compiles the code without making any changes to the organization (`true`) or compiles and saves the code (`false`). This field is required. **Note:** You can compile without saving but you can't save without compiling.
`IsRunTests`	**Type** boolean **Properties** None **Description** Reserved for future use.
`MetadataContainerId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** The ID of a MetadataContainer object. Specify a `MetadataContainerId` or a `MetadataContainerMemberId`, but not both.
`MetadataContainerMemberId`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort **Description** The ID of an ApexClassMember, ApexTriggerMember, ApexPageMember or ApexComponentMember object. Specify a `MetadataContainerId` or a `MetadataContainerMemberId`, but not both.
`State`	**Type** picklist **Properties** Filter, Group, Restricted picklist, Sort

Field Name	Details
	Description The state of the request. Valid values are: • `Queued`—the job is in the queue. • `Invalidated`—Salesforce cancelled the job because the results might not be valid. This state occurs if someone changes the container members while `IsCheckOnly=true`, or if a newer compile request is added to the queue. • `Completed`—the compilation or deployment finished. The `SymbolTable` fields for the specified object(s) were successfully updated. If `IsCheckOnly` is `false`, the `Body` for each object was saved and the `MetadataContainerId` field for each object was reset from the ID of the deployed MetadataContainer to the ID of the corresponding ContainerAsyncRequest object. • `Failed`—the compilation or deployment failed for the reasons stated in the `CompilerError` field. • `Error`—an unexpected error occurred. The messages in the `ErrorMsg` field can be provided to Salesforce support if the issue persists. • `Aborted`—use this value to delete a queued deployment. This field is required.

Usage

When you deploy a ContainerAsyncRequest, you must specify whether to save the compiled entities:

- To compile entities without saving, set the request to `IsCheckOnly=true`. This option is only supported if a `MetadataContainerMember` is specified. A single `MetadataContainerMemberId` can't be compiled without saving.
- To compile and save entities to your organization, set the request to `IsCheckOnly=false`.

If the compile succeeds, the `SymbolTable` field is updated on each object in the specified MetadataContainer. If the save or compile fails and a `SymbolTable` field cannot be updated, the field is cleared. If there is an outstanding save request, all updates, inserts, and deployments fail.

To terminate a queued deployment, set the `State` field to `Aborted`.

CustomField

Represents a custom field on a custom object that stores data unique to your organization. Includes access to the associated CustomField object and related fields in Salesforce Metadata API.

Available from API version 28.0 or later.

Supported SOAP Calls

`create()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST HTTP Methods

`Query`, `GET`, `POST`, `PATCH`

Fields

Field Name	Details
`DeveloperName`	**Type** string **Properties** Create, Filter, Group, Sort, Update **Description** The developer's internal name for the custom field (for example "CF_c").
`Metadata`	**Type** CustomFieldMetadata **Properties** Create, Update, Nillable **Description** CustomFieldMetadata includes the following fields: • `caseSensitive` • `customDataType` • `defaultValue` • `deleteConstraint` • `deprecated` • `description` • `displayFormat` • `displayLocationInDecimal` • `escapeMarkup` • `externalDeveloperName` • `externalId` • `formula` • `formulaTreatBlanksAs` • `inlineHelpText` • `isFilteringDisabled` • `isNameField` • `isSortingDisabled` • `label` • `length` • `maskChar` • `maskType` • `picklist` • `populateExistingRows` • `precision` • `readOnlyProxy` • `referenceTo` • `relationshipLabel` • `relationshipName`

Field Name	Details
	• relationshipOrder • reparentableMasterDetai • required • restrictedAdminField • scale • startingNumber • stripMarkup • summarizedField • summaryFilterItems • summaryForeignKey • summaryOperation • trackFeedHistory • trackHistory • type • unique • visibleLines • writeRequiresMasterRead
NamespacePrefix	**Type** string **Properties** Filter, Group, Sort, Nillable **Description** The namespace of the custom field. A custom field can be in an extension namespace different than the object.
TableEnumOrId	**Type** picklist **Properties** Create, Filter, Group, Sort **Description** The enum (for example, Account) or ID of the object this field is on.

CustomObject

Represents a custom object that stores data unique to your organization. Includes access to the associated CustomObject object and related fields in Salesforce Metadata API.

Available from API version 31.0 or later.

Supported SOAP Calls

`query()`, `search()`, `retrieve()`

Supported REST HTTP Methods

`Query`, `GET`

Fields

Field Name	Details
`DeveloperName`	**Type** string **Properties** Filter, Group, Sort **Description** The developer's internal name for the custom object (for example "CF_c").
`ExternalRepository`	**Type** string **Properties** Filter, Group, Sort, Nillable **Description** Maps to a table in the external data source. If you created the external object when using Validate and Sync for the data source, this name is automatically created; do not modify it.
`ExternalName`	**Type** string **Properties** Filter, Group, Sort, Nillable **Description** Maps to a table in the external data source. If you created the external object using Validate and Sync for the data source, this name is automatically created.
`NamespacePrefix`	**Type** string **Properties** Filter, Group, Sort, Nillable **Description** The namespace of the custom object.

DeployDetails

A complex type that contains detailed XML for any compile errors reported in the asynchronous request defined by a ContainerAsyncRequest object.

Replaces the JSON field `CompilerErrors` in Tooling API version 31.0 and later.

Fields

Field	Details

`componentFailures`	**Type**
	string
	
	Description
	The line number, component name and a short description for any compile errors. For example:
	
	`<DeployDetails>`
	`  <componentFailures>`
	`  <lineNumber>5</lineNumber>`
	`  <fullName>myApex</fileName>`
	`  <problem>invalid name 'abc'</problem>`
	`  </componentFailures>`
	`  <componentFailures>`
	`  <lineNumber>10</lineNumber>`
	`  <fullName>myApex2</fileName>`
	`  <problem>invalid type 'hello'</problem>`
	`  </componentFailures>`
	`</DeployDetails>`

HistoryRetentionJob

Represents the body of retained data from the archive, and the status of the archived data. Available in API version 29.0 or later.

Note: The HistoryRetentionJob object is currently available through a limited pilot program. Contact your salesforce.com representative to see if your organization qualifies.

Supported SOAP API Calls

`describeSObjects(), query()`

Supported REST API HTTP Methods

`GET`

Fields

Field Name	Details
`CreatedById`	**Type** reference **Properties** Defaulted on create, Filter, Group, Sort **Description** The user ID of the user who created the field history retention job.
`CreatedDate`	**Type** dateTime **Properties** Defaulted on create, Filter, Sort **Description** The date the record was saved.
`DurationSeconds`	**Type** int **Properties** Filter, Group, Nillable, Sort **Description** How many seconds the field history retention job took to complete (either successfully or with failures).
`HistoryType`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Restricted picklist, Sort **Description** The object type containing the field history you retained. Valid values are: • `Account` • `Case` • `Contact` • `Leads` • `Opportunity` Custom objects are also valid.
`NumberOfRowsRetained`	**Type** int

Field Name	Details
	Properties Filter, Group, Nillable, Sort **Description** The number of field history rows a field history retention job has retained.
`RetainOlderThanDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date and time before which all field history data was retained.
`StartDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The start date of the field history retention job.
`Status`	**Type** picklist **Properties** Filter, Group, Nillable, Restricted picklist, Sort **Description** Provides the status of the field history retention job. By default, the pilot feature copies data to the archive, leaving a duplicate of the archived data in Salesforce. Deletion of data from Salesforce after archiving is available upon request. Status may include: • `CopyScheduled` • `CopyRunning` • `CopySucceeded` • `CopyFailed` • `CopyKilled` • `DeleteScheduled` • `DeleteRunning` • `DeleteSucceeded` • `DeleteFailed` • `DeleteKilled`

FlexiPage

Represents a Flexible Page. A Flexible Page is the home page for an app that appears as a menu item in the Salesforce1 navigation menu. Includes access to the associated FlexiPage object in the Salesforce Metadata API.

Available from API version 31.0 or later.

Supported SOAP Calls

`create()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST HTTP Methods

`GET`, `HEAD`

Fields

Field	Details
`DeveloperName`	**Type** string **Properties** Create, Filter, Group, Sort, Update **Description** The API name of the Flexible Page.
`FullName`	**Type** string **Properties** Group, Nillable. **Description** The full name of the associated FlexiPage object in Metadata API.
`LastModifiedDate`	**Type** dateTime **Properties** Filter, Sort. **Description** Last modified date.
`Metadata`	**Type** FlexiPageMetadata **Properties** Create, Nillable, Update.

Field	Details
	Description Flexible Page metadata.
NamespacePrefix	**Type** string **Properties** Filter, Group, Nillable, Sort. **Description** The namespace prefix.
Type	**Type** picklist **Properties** Filter, Group, Restricted picklistSort. **Description** Required. The type of the Flexible Page This field is available in API version 32.0 or later. In API version 32.0, this field can only have a value of AppPage.

Sample Code

This code sample creates a new Flexible Page with a single Recent Items component, that shows recently used Accounts and `MyCustomObject__cs`

```
 ComponentInstance recentItems = new ComponentInstance();
 recentItems.setComponentName("flexipage:recentItems");
 ComponentInstanceProperty cip = new ComponentInstanceProperty();
 cip.setName("entityNames");
 cip.setValue("Account,MyCustomObject__c");
 recentItems.setComponentInstanceProperties(new ComponentInstanceProperty[]{cip});

FlexiPageRegion mainRegion = createRegion("main");
mainRegion.setComponentInstances(new ComponentInstance[] { recentItems });

FlexiPageMetadata fpMetadata = new FlexiPageMetadata();
fpMetadata.setFlexiPageRegions(new FlexiPageRegion[]{mainRegion});
fpMetadata.setMasterLabel("My FlexiPage");
fpMetadata.setDescription("A FlexiPage with a recent items component");

FlexiPage flexiPage = new FlexiPage();
flexiPage.setFullName("MyFlexiPageDevName");
flexiPage.setMetadata(fp);

// Create
SaveResult saveResult = soapConnection.create(new SObject[] { flexiPage });
```

HeapDump

A complex type that represents a heap dump in an ApexExecutionOverlayResult.

Available from API version 28.0 or later.

Fields

Field	Details
`className`	**Type** string **Description** The name of the Apex class or trigger.
`extents`	**Type** array of TypeExtent **Description** TypeExtent includes the following fields: • `collectionType` • `count` • `definition` (array of AttributeDefinition) • `extent` (array of HeapAddress) • `totalSize` • `typeName`
`heapDumpDate`	**Type** dateTime **Description** The date and time that the heap dump was captured.
`namespace`	**Type** string **Description** The namespace of the Apex class or trigger. Null if there is no namespace.

Usage

Use heap dumps to capture structured debugging information.

MenuItem

Represents a menu item.

This object is available in API version 32.0 and later.

Supported SOAP Calls

`query()`, `update()`

Supported REST HTTP Methods

`GET`, `POST`

Fields

Field	Details
`Active`	**Type** boolean **Properties** Defaulted on create, Filter, Group, Sort, Update **Description** Indicates whether the item in the menu is active (`true`) or not (`false`).
`AppId`	**Type** string **Properties** Filter, Group, Sort **Description** The ID of the app that this menu item is associated with. Can be an enum (such as Feed or People) or an alphanumeric ID. Use `AppId` as the unique ID for the menu item, not `Id`.
`Color`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The color of the menu item that appears in the user interface. This field is described in Web color RGB format, such as `00FF00`.
`IconURL`	**Type** url **Properties** Filter, Group, Nillable, Sort **Description** The URL of an icon in the menu item.

Field	Details
`Label`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The menu item label that appears in the user interface.
`MenuType`	**Type** picklist **Properties** Filter, Group, Nillable, Restricted picklist, Sort **Description** The type of menu that this menu item belongs to. Valid values are: • `AppSwitcher`: the Force.com app menu, a drop-down menu that's displayed at the top of every app page • `Salesforce1`: the Salesforce1 navigation menu • `NetworkTabs`: the Salesforce Communities tab set This field is required for `query()`.
`SortOrder`	**Type** int **Properties** Filter, Group, Nillable, Sort, Update **Description** The `SortOrder` value determines the order in which a menu item is displayed in the user interface. This field must be an ordinal number greater than 0, and must be unique in the list. Inactive menu items have a value of -1.
`Theme`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The associated theme, which must be one of the following values. • `theme2`: the Salesforce theme that was used prior to Spring '10 • `theme3`: the Salesforce theme that was introduced in Spring '10 • `theme4`: the theme that was introduced in Winter '14 for the mobile touchscreen version of Salesforce • `custom`: the theme that's associated with a custom icon

Field	Details
	This field is required for `query()` for `Color` and `IconURL`.

Usage

MenuItem can be queried and manipulated to change how menu items appear in Salesforce. The following example modifies the Salesforce1 left navigation menu.

```
String query = "SELECT AppId, Label, Active, SortOrder FROM MenuItem "
+
     "WHERE MenuType = 'Salesforce1'";
SObject[] records = sforce.query(query).getRecords();

//Activate all menu items
for (int i = 0; i < records.length; i++) {
   MenuItem item = (MenuItem)records[i];
   item.setOrder(i + 1);
   item.setActive(true);
}

sforce.update(records);
```

MetadataContainer

Manages working copies of ApexClassMember, ApexTriggerMember, ApexPageMember and ApexComponentMember objects, including collections of objects that should be deployed together.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`Name`	**Type** string **Properties** Create, Filter, Group, Sort, Update **Description** The name of the MetadataContainer. If a container with the same name already exists, an error is reported on `create()` or `POST`. This field is required.

Usage

Use a MetadataContainer as a package for your tool's workspace. As a user works in the tool, update the ApexClassMember, ApexTriggerMember, ApexPageMember and ApexComponentMember objects in the MetadataContainer and use a ContainerAsyncRequest object to save and deploy changes to the current organization.

A MetadataContainer can be reused, but container members can't.

- When a ContainerAsyncRequest completes successfully, the `MetadataContainerId` field on all container members is changed from the ID of the MetadataContainer to the ID of the ContainerAsyncRequest. At this point, container members can no longer be modified or deployed, and can't be queried via the MetadataContainer; you have to query the ContainerAsyncRequest to see what was deployed.
- If the deployment fails, container members remain on the MetadataContainer and can still be modified until they are successfully deployed on another ContainerAsyncRequest. The `MetadataContainerId` field on the completed (failed deployment) ContainerAsyncRequest is set to the ID of the MetadataContainer, so you can have multiple completed ContainerAsyncRequests on a single MetadataContainer.

Note: Deleting a MetadataContainer deletes all objects that reference it.

See Also:

Metadata API Developer's Guide

QuickActionDefinition

Represents the definition of a quick action.

This object is available in API version 32.0 and later.

Supported SOAP Calls

`create(), delete(), query(), retrieve(), update(), upsert()`

Supported REST HTTP Methods

`DELETE`, `GET`, `PATCH`, `POST`

Fields

Field	Details
`Description`	**Type** textarea **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** The description of the action.
`DeveloperName`	**Type** string

Field	Details
	Properties Create, Filter, Group, Sort, Update **Description** The unique name of the action in the API. This field corresponds to the **Name** field in the user interface.
`Height`	**Type** int **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** The height of the action, in pixels. This field is set only when the quick action has a custom icon.
`IconId`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** The ID of the action icon. This field is set only when the quick action has a custom icon.
`Label`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The action label that corresponds to the **Label** field in the user interface.
`Language`	**Type** picklist **Properties** Create, Defaulted on create, Filter, Group, Nillable, Restricted picklist, Sort, Update **Description** The language of the action. Valid values are: • Chinese (Simplified): `zh_CN` • Chinese (Traditional): `zh_TW` • Danish: `da` • Dutch: `nl_NL` • English: `en_US`

Field	Details
	• 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`
`MasterLabel`	**Type** string **Properties** Create, Filter, Group, Sort, Update **Description** The action label.
`NamespacePrefix`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The namespace of the action.
`SobjectType`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort **Description** The associated object's API name. For example, `FeedItem`.
`StandardLabel`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Restricted picklist, Sort, Update

Field	Details
	Description The standard label for the action. Valid values are: • `ChangeDueDate` • `ChangePriority` • `ChangeStatus` • `CreateNew` • `CreateNewRecordType` • `Defer` • `EditDescription` • `LogACall` • `LogANote` • `New` • `NewChild` • `NewChildRecordType` • `NewRecordType` • `Quick` • `QuickRecordType` • `SendEmail` • `SocialPost` • `Update`
`TargetField`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Restricted picklist, Sort, Update **Description** The API name of the parent object for the record created by this quick action. For example, `CollaborationGroup`.
`TargetRecordTypeId`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** The ID of the target record type.
`TargetSobjectType`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Restricted picklist, Sort, Update

Field	Details
	Description The API name of the type of object record this action will create. For example, `OpportunityLineItem`.
`Type`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The type of action. Valid values are: • `Canvas` • `Create` • `LogACall` • `Post` • `SendEmail` • `SocialPost` • `Update` • `VisualforcePage`
`Width`	**Type** int **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** The width of the action, in pixels. This field is set only when the quick action has a custom icon.

Usage

A QuickActionDefinition represents information about a quick action. The following example creates a global quick action that lets users quickly create a task.

```
QuickActionDefinition qad = new QuickActionDefinition();
qad.setDeveloperName("MyQuickCreateTaskAction");
qad.setSobjectType("Global");
qad.setTargetSobjectType("Task");
qad.setMasterLabel("Quick create a task");
qad.setType(QuickActionType.Create);
qad.setDescription("Quickly creates a Task");

sforce.create(new SObject[]{qad});
```

QuickActionList

Represents a list of quick actions.

This object is available in API version 32.0 and later.

Supported SOAP Calls

`create(), query(), retrieve(), update(), upsert()`

Supported REST HTTP Methods

`DELETE, GET, PATCH, POST`

Fields

Field	Details
`LayoutId`	**Type** reference **Properties** Create, Filter, Group, Sort **Description** The ID of the associated layout.

Usage

A QuickActionList is a junction between QuickActionListItem objects and a layout. If a layout doesn't have an associated QuickActionList, it inherits the actions from the global page layout.

The following example retrieves all quick action lists in an organization and their associated layout ID.

```
String query = "SELECT Id,LayoutId FROM QuickActionList";
SObject[] records = sforce.query(query).getRecords();

for (int i = 0; i < records.length; i++) {
   QuickActionList list = (QuickActionList)records[i];
   String relatedLayoutId = list.get("LayoutId");
}
```

QuickActionListItem

Represents an item in a quick action list.

This object is available in API version 32.0 and later.

Supported SOAP Calls

`create(), delete(), query(), retrieve(), update(), upsert()`

Supported REST HTTP Methods

`DELETE, GET, PATCH, POST`

Fields

Field	Details
QuickActionDefinition	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The enum name or ID of the QuickActionDefinition that's associated with this list item. Valid values are: • Case.ChangeStatus • Case.LogACall • FeedItem.ContentPost • FeedItem.LinkPost • FeedItem.MobileSmartActions • FeedItem.PollPost • FeedItem.QuestionPost • FeedItem.TextPost
QuickActionListId	**Type** reference **Properties** Create, Filter, Group, Sort **Description** The ID of the QuickActionList associated with this list item.
SortOrder	**Type** int **Properties** Create, Filter, Group, Sort, Update **Description** The order in which this list item appears in the picklist. This field must be an ordinal number greater than 0, and must be unique in the list.

Usage

A QuickActionListItem associates a QuickActionDefinition with a QuickActionList. You can query to find out which quick actions are in a list, insert or delete to add or remove quick actions from a list, and update to change the order of quick actions in the list.

The following example reverses the order in the list of the actions, and then removes the first action from the list.

```
String query = "SELECT Id,SortOrder FROM QuickActionListItem Where QuickActionListId='" +
 listId + "'"
SObject[] records = sforce.query(query).getRecords();
```

```
for(int i=0;i<records.length;i++) {
   QuickActionListItem item = (QuickActionListItem)records[i];
   item.setSortOrder(records.length-i);
}

sforce.update(records);

// Last record in array is first record in reordered list
sforce.delete(records[records.length-1].getId());
```

SOQLResult

A complex type that represents the result of a SOQL query in an ApexExecutionOverlayResult.

Available from API version 28.0 or later.

Fields

Field	Details
queryError	**Type** string **Description** The error text returned if the execution was unsuccessful.
queryMetadata	**Type** QueryResultMetadata **Description** The structured result returned from a successful execution. QueryResultMetadata includes the following fields: • columnMetadata • entityName • groupBy • idSelected • keyPrefix
queryResult	**Type** array of MapValue **Description** MapValue contains an array of MapEntry, which contains the following fields: • keyDisplayValue • value (reference to StateValue)

Usage

Overlay SOQL on checkpoints to capture structured debugging information.

StaticResource

Represents the working copy of a static resource file for editing or saving. Static resources allow you to upload content that you can reference in a Visualforce page, including images, stylesheets, JavaScript, and other files.

Available in Tooling API version 29.0 and later.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`Name`	**Type** string **Properties** Create, Update **Description** 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
`Body`	**Type** string **Properties** Create, Update **Description** The data for the static resource file.
`ContentType`	**Type** string **Properties** Create, Update **Description** Required. The content type of the file, for example text/plain.
`CacheControl`	**Type** string

Field Name	Details
	Properties Create, Update **Description** Required. Indicates whether the static resource is marked with a public caching tag so that a third-party delivery client can cache the content. The valid values are: • Private • Public

Usage

To create, edit, or save a static resource file, create a StaticResource object that references it.

SymbolTable

A complex type that represents all user-defined tokens in the `Body` of an ApexClass, ApexClassMember, or ApexTriggerMember and their associated line and column locations within the `Body`.

Fields

Field	Details
`constructors`	**Type** array of Constructor **Description** Contains the position, scope and signature of constructors for the Apex class. Apex triggers don't have constructors. Constructor includes the following fields: • `location` • `modifiers` • `name` • `references` • `visibility` (scope: Global, Public or Private) • `parameters`
`externalReferences`	**Type** array of ExternalReference **Description** Contains the name, namespace, and external class, method and variable references for the Apex class or trigger. These references can be used for symbol highlighting or code navigation.

Field	Details
	ExternalReference includes the following fields: • `methods` • `modifiers` • `name` • `namespace` • `references` • `variables`
`innerClasses`	**Type** array of SymbolTable **Description** Contains a symbol table for each inner class of the Apex class or trigger.
`interfaces`	**Type** array of String **Description** Contains a set of strings for each interface with the namespace and name, for example: `['System.Batchable', 'MyNamespace.MyInterface']`.
`methods`	**Type** array of Method **Description** Contains the position, name, scope, signature, and return type of available Apex methods. Method includes the following fields: • `location` • `modifiers` • `name` • `references` • `visibility` (scope: Global, Public or Private) • `parameters` • `returnType`
`name`	**Type** string **Description** The name of the Apex class or trigger.
`namespace`	**Type** string

Field	Details
	Description The namespace of the Apex class or trigger. Null if there is no namespace.
`properties`	**Type** array of VisibilitySymbol **Description** Contains the position, name, scope, and references of properties for the Apex class or trigger. VisibilitySymbol includes the following fields: • `location` • `modifiers` • `name` • `references` • `visibility` (scope: Global, Public or Private)
`tableDeclaration`	**Type** array of Symbol **Description** Contains the position, name, and references of the Apex class or trigger. Symbol includes the following fields: • `location` • `modifiers` • `name` • `references`
`variables`	**Type** array of Symbol **Description** Contains the position, name and references of related variables. Symbol includes the following fields: • `location` • `modifiers` • `name` • `references`

Usage

Use symbol tables instead of building a parser or compiler. Symbol tables allow you to do symbol highlighting, code navigation, code completion, symbol searches, and more.

A symbol table cannot be created if the content referenced by the `ContentEntityId` field doesn't use a symbol table, or if compiler errors for the last deployment of the MetadataContainer in the `MetadataContainerId` field prevented a symbol table from being created.

TraceFlag

Represents a trace flag that triggers an Apex debug log at the specified logging level.

Supported SOAP API Calls

`create()`, `delete()`, `describeSObjects()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST API HTTP Methods

`Query`, `GET`, `POST`, `PATCH`, `DELETE`

Fields

Field Name	Details
`ApexCode`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category level for Apex code. Includes information about Apex code and can include information such as log messages generated by data manipulation language (DML) statements, inline SOQL or SOSL queries, the start and completion of any triggers, the start and completion of any test method, and so on. The following are valid values. • `Finest` • `Finer` • `Fine` • `Debug` • `Info` • `Warn` • `Error` This field is required.
`ApexProfiling`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category level for profiling information. Includes cumulative profiling information, such as the limits for your namespace, the number of emails sent, and so on. The following are valid values. • `Finest` • `Finer` • `Fine`

Field Name	Details
	• `Debug` • `Info` • `Warn` • `Error` This field is required.
`Callout`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category level for callouts. Includes the request-response XML that the server is sending and receiving from an external Web service. This is useful when debugging issues related to SOAP API calls. The following are valid values. • `Finest` • `Finer` • `Fine` • `Debug` • `Info` • `Warn` • `Error` This field is required.
`Database`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category for database activity. Includes information about database activity, including every DML statement or inline SOQL or SOSL query. The following are valid values. • `Finest` • `Finer` • `Fine` • `Debug` • `Info` • `Warn` • `Error` This field is required.

Field Name	Details
`ExpirationDate`	**Type** dateTime **Properties** Create, Filter, Sort, Update **Description** The date and time that the trace flag expires. This field is required.
`ScopeId`	**Type** reference **Properties** Create, Filter, Group, Nillable, Sort, Update **Description** A reference to a user. This field is used with the `TracedEntityID` field. • When `ScopeId=user` the actions of the user/entity specified by `TracedEntityID` (user, Apex class or Apex trigger) are traced to the system log at the described level. System logs are visible only to you. Use this scope for class-level filtering. If there are both user and entity-level flags, the user flags take precedence until a method from a class with an entity trace flag is entered. When the method returns, the user trace flags are restored. • When `ScopeId=`***`emptyid`*** the user's actions are traced to the organization's debug log at the described level. Debug logs are visible to all administrators. This option is only available if `TracedEntityID` references a user (not an Apex class or Apex trigger). The variable *`emptyid`* can be the value 0000000000000000 or null. The scope defined here is reflected in the ApexLog `Location` field.
`System`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category level for calls to all system methods, such as the `System.debug` method. The following are valid values. • `Finest` • `Finer` • `Fine` • `Debug` • `Info` • `Warn` • `Error` This field is required.

Field Name	Details
`TracedEntityId`	**Type** reference **Properties** Create, Filter, Group, Sort, Update **Description** A reference to the following: • Apex class • Apex trigger • User This field is used with the `ScopeId` field. This field is required.
`Validation`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category level for validation rules. Includes information about validation rules, such as the name of the rule, whether the rule evaluated true or false, and so on. The following are valid values. • `Finest` • `Finer` • `Fine` • `Debug` • `Info` • `Warn` • `Error` This field is required.
`Visualforce`	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category level for Visualforce. Includes information about Visualforce events, including serialization and deserialization of the view state or the evaluation of a formula field in a Visualforce page. The following are valid values. • `Finest` • `Finer` • `Fine`

Field Name	Details
	• Debug • Info • Warn • Error This field is required.
Workflow	**Type** picklist **Properties** Create, Filter, Group, Restricted picklist, Sort, Update **Description** The log category level for workflow rules. Includes information for workflow rules, such as the rule name, the actions taken, and so on. This field is required. The following are valid values. • Finest • Finer • Fine • Debug • Info • Warn • Error

Usage

To diagnose a functional issue or a performance problem, use the TraceFlag object to set up logging for yourself or for another user. The following options are available:

- To set up a log for a specific user, set `ScopeId` to null and `TracedEntityId` to the ID of the user. This option can only be configured for a user, not an Apex class or Apex trigger.
- To configure logging levels for system logs (visible only to you), set `ScopeId` to `user` and `TracedEntityId` to the ID of the logged-in user.
- To set up a system log (visible only to you) for a specific Apex class or trigger, set `ScopeId` to `user` and `TracedEntityId` to the ID of the Apex class or trigger.

The example below creates a new trace flag and attaches it to an end user.

```
//create a new TraceFlag object
TraceFlag traceFlag = new TraceFlag();
traceFlag.ApexCode = "Finest";
traceFlag.ApexProfiling = "Finest";
traceFlag.Callout = "Info";
traceFlag.Database = "Finest";
traceFlag.System = "Debug";
traceFlag.Validation = "Info";
traceFlag.Visualforce = "Info";
traceFlag.Workflow = "Info";

//set an expiration date
traceFlag.ExpirationDate = myTimestamp;
//set the ID of the user to monitor
```

```
traceFlag.TracedEntityId = "005A0000000i93O";

//call the create method
TraceFlag[] traceFlags = { traceFlag };
SaveResult[] traceResults = sforce.create(traceFlags);
for (int i = 0; i < traceResults.Length; i++)
{
   if (traceResults[i].success)
   {
      Console.WriteLine("Successfully created trace flag: " +
      traceResults[i].id);
   }
   else
   {
      Console.WriteLine("Error: could not create trace flag ");
      Console.WriteLine("   The error reported was: " +
      traceResults[i].errors[0].message + "\n");
   }
}
```

ValidationRule

Represents a formula that is used for specifying when a criteria is met. This includes both validation rules and workflow rules. Includes access to the associated ValidationRule object in the Salesforce Metadata API.

Available from API version 30.0 or later.

Supported SOAP Calls

`create()`, `delete()`, `query()`, `retrieve()`, `update()`, `upsert()`

Supported REST HTTP Methods

`Query`, `GET`, `POST`, `PATCH`

Fields

Field Name	Details
`FullName`	**Type** string **Properties** Group, Nillable. **Description** The full name of the associated object in the Metadata API.
`Id`	**Type** id **Properties** Filter, Group, Sort. **Description** The ID of a specific validation rule.

Field Name	Details
`LastModifiedDate`	**Type** dateTime **Properties** Filter, Sort. **Description** Last modified date.
`Metadata`	**Type** ValidationRuleMetadata **Properties** Create, Nillable, Update. **Description** Validation rule metadata.
`NamespacePrefix`	**Type** string **Properties** Filter, Group, Nillable, Sort. **Description** The namespace prefix.
`TableEnumOrId`	**Type** picklist **Properties** Create, Filter, Group, Sort **Description** The enum (for example, Account) or ID of the object this field is on.
`ValidationName`	**Type** string **Properties** Create, Filter, Group, Sort, Update. **Description** The enum name or ID of entity this rule is associated with.

WorkflowRule

Represents a workflow rule that is used to fire off a specific workflow action when the specified criteria is met. Includes access to the associated WorkflowRule object in Salesforce Metadata API.

Available from API version 30.0 or later.

Supported SOAP Calls

`create(), query(), retrieve()`

Supported REST HTTP Methods

`Query, GET`

Fields

Field Name	Details
`FullName`	**Type** string **Properties** Group, Nillable. **Description** The Metadata API full name.
`Id`	**Type** id **Properties** Filter, Group, Sort. **Description** The ID of a specific workflow rule.
`LastModifiedDate`	**Type** datetime **Properties** Filter, Sort. **Description** Last modified date.
`Metadata`	**Type** WorkflowRuleMetadata **Properties** Nillable.

Field Name	Details
	Description Workflow rule metadata.
`Name`	**Type** string **Properties** Filter, Group, Sort. **Description** The enum name or ID of entity this rule is associated with.
`NamespacePrefix`	**Type** string **Properties** Filter, Group, Nillable, Sort. **Description** The namespace prefix.
`TableEnumOrId`	**Type** picklist **Properties** Filter, Group, Sort. **Description** The enum (for example, Account) or ID of the object this field is on.

Index

A

Apex
- Debugging 22, 24, 27, 70
- Deploying 43, 57
- Editing 12–13, 30, 40
- Saving 12, 30, 40
- Saving and compiling 13, 40, 43, 57
- Viewing code coverage 16, 18, 29

ApexClass object 12
ApexClassMember object 13
ApexCodeCoverage object 16
ApexCodeCoverageAggregate object 18
ApexComponent object 19
ApexComponentMember object 20
ApexExecutionOverlayAction object 22
ApexExecutionOverlayResult object 24
ApexLog object 27
ApexOrgWideCoverage object 29
ApexPage object 30
ApexPageMember object 30
ApexResult object 32
ApexTestQueueItem object 33
ApexTestResult object 37
ApexTrigger object 40
ApexTriggerMember object 40

C

Checkpoint 32, 54, 65
Compile errors 49
ContainerAsyncRequest object 43
CSS
- Editing 66

CustomField object 45
CustomObject object 47

D

Debugging 27, 70
Debugging Apex 22, 24
DeployDetails object 49
Deploying Apex 43, 57
Deploying Visualforce 43, 57
Developer Console 1

E

Editing Apex 12–13, 30, 40
Editing Visualforce 19–20, 30

F

FlexiPage object 52

G

Generating heap dumps 22, 24, 27

H

Heap dump 54
Heap dumps 22, 24, 27
HeapDump object 54
HistoryRetentionJob object 49

J

JavaScript
- Editing 66

L

Log 32, 54, 65
Logging 27, 70

M

MenuItem object 54
MetadataContainer object 57

O

Objects
- ApexClass 12
- ApexClassMember 13
- ApexCodeCoverage 16
- ApexCodeCoverageAggregate 18
- ApexComponent 19
- ApexComponentMember 20
- ApexExecutionOverlayAction 22
- ApexExecutionOverlayResult 24
- ApexLog 27
- ApexOrgWideCoverage 29
- ApexPage 30
- ApexPageMember 30
- ApexResult 32
- ApexTestQueueItem 33
- ApexTestResult 37
- ApexTrigger 40
- ApexTriggerMember 40
- ContainerAsyncRequest 43
- CustomField 45
- CustomObject 47
- DeployDetails 49
- FlexiPage 52
- HeapDump 54
- HistoryRetentionJob 49
- MenuItem 54

Objects *(continued)*
- MetadataContainer 57
- QuickActionDefinition 58
- QuickActionList 62
- QuickActionListItem 63
- SOQLResult 65
- StaticResource 66
- SymbolTable 67
- TraceFlag 70
- ValidationRule 75
- WorkflowRule 77

Overview 1

Q

QuickActionDefinition object 58
QuickActionList object 62
QuickActionListItem object 63

R

REST API 2

S

Saving and compiling Apex 13, 40, 43, 57
Saving and compiling Visualforce 20, 30, 43, 57
SOAP API 5
SOQLResult object 65
Standard objects 10
StaticResource object 66
Symbol tables 67
SymbolTable object 67

T

Tasks 1
Tests 33, 37
TraceFlag object 70

V

ValidationRule object 75
Visualforce
- Deploying 43, 57
- Editing 19–20, 30
- Saving and compiling 19–20, 30, 43, 57

W

WorkflowRule object 77

X

XML
- Editing 66