mentor.axeda.com

Tuesday, February 23, 2016 9:22 AM

Custom Object Chaining

Tuesday, February 23, 2016 9:22 AM

When building complex solutions, it is often necessary to divide your logic into separate Custom Objects. Unlike the Object Oriented model of invoking methods on classes, Custom Objects can be "chained" together by having one Custom Object invoke another. This model allows you to pass Maps of Parameters just as you would do when invoking Custom Objects from Expression Rules or via Scripto.

There are two ways to invoke a Custom Object from within another Custom Object: Synchronously and Asynchronously. This Getting Started will walk you through a quick example of each approach.

Sync Chaining

Sync_co.jpg

In Synchronous mode you will be making a "blocking" call from one Custom Object to another (in this example, the "Caller" CO is calling the "Called" CO). This is important, as the request will stay in memory for the duration of the Call's execution.

The request is made by passing a Map of <String, Object> to the Called Custom Object. In return, the Called Custom Object is going to return a java.lang.Object. This can be any Object, but it will have to be cast after it is returned.

In the example below, we declare the CO to be Called ("addFive") and the CO that will call it ("returnsTwenty").

addFive

```
import com.axeda.drm.sdk.customobject.Call

// add five to the input value
return (Call.parameters.theNumber + 5)
```

returnsTwenty

```
import com.axeda.sdk.v2.dsl.Bridges

// call the first custom object
def twenty = Bridges.customObjectBridge.execute("addFive", [theNumber: 15]) as Integer

// log it for posterity
logger.info twenty

// return the value as-is
return twenty
```

To invoke a Custom Object, we use the CustomObjectBridge.execute() method. This will then populate the "Call" Parameters, pass it into the Called CO, and return the output as a big "O" Object.

This is important, as the return value will always be type erased, so you will need to cast or coerce it to

your desired type.

In the "addFive" CO you see the "Call" Object which is getting passed in during the execution. This is ALWAYS the Object we will use during Custom Object chaining to receive our variables. The Call Object will provide the list of Parameters that are sent into the Custom Object. You can access these Parameters as a Map from Call.parameters.

Async Chaining

Callback.png

There are also ways to call Custom Objects Asynchronously. This is useful for calling external web services that may take a long time to return, for doing work in batches, or when you are concerned about execution time.

This is specifically useful when calling a Custom Object from an Expression Rule.

This example runs like a "Thread" on the Platform by invoking itself and incrementing a counter until the count reaches 10. This is not "Recursion", since the requests are all discreet, and not made on the same stack.
Use Bridges.customObjectBridge.initiateAsync() to request an Async invocation.

countToTenThread

```
//name: addToTen
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.drm.sdk.customobject.Call
import com.axeda.services.v2.ExecutionResult

def count = Call.parameters.theNumber

if(count == null){
  //Initial iteration.  Start at 1.
  logger.info "Start Run!"
  count = 1
}

if(count >= 10) {
  logger.info "All Done!"
} else {
  logger.info("Current Number: ${count}")
  //Call recursively call this Custom Object with a 10 second timeout
  ExecutionResult er = Bridges.customObjectBridge.initiateAsync("countToTenThread", [theNumber:
count +1], 10)
  println "Id: " +er.succeeded.getAt(0).id
}
```

Once again you see that the "Call" Object which is getting passed in during the execution. Because the Async invocations need to be made via a queue, the parameters MUST be serializable (the Call will actually contain a Map of String,Serializable).

Also, the result of an Async initiation will always be a standard ExecutionResult object. This will contain the success/failure of the Request (NOT of the Execution), and will contain the UUID that you can use to

lookup the status and the result.

Async Results

Callback.png

One of the biggest benefits of the Async Chaining approach is that the results of the Async execution will be stored for future retrieval. This allows us to perform some operation asynchronously, and then collect the output at a later date. The maximum size of a stored result is 65536 bytes. This is also the limit for the size of the Parameters map which is passed into an Async invocation.

The following code example shows the Token / Poll approach of initiating an Async job and then using the Token returned from the initiateAsync call to look up the results afterwards.

waitForExecution

```
//name: waitForExecution
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.drm.sdk.customobject.Call
import com.axeda.services.v2.AsyncCustomObjectExecutionStatus
import com.axeda.services.v2.ExecutionResult

def message = "This is our message"

// call our string reverse co
ExecutionResult er = Bridges.customObjectBridge.initiateAsync("reverseString", [message: message],
100)

// find the uuid for the Async execution, we will use this to look up the result
def uuid = UUID.fromString(er.succeeded.get(0).id)

// now look up the status until it is completed
int limit = 100
for (i in 1..limit) {
   AsyncCustomObjectExecutionStatus status = Bridges.customObjectBridge.getAsyncStatus(uuid)
   if (["SUCCESS","FAILED"].contains(status.value())) {
      // it eaither succeeded or it failed, so return whatever we got back
      return Bridges.customObjectBridge.getAsyncResult(uuid)
   } else {
      Thread.sleep(1000)
   }
}

// should not reach here! But, exit cleanly anyway
return "Async Job not completed in time"
```

reverseString

```
import com.axeda.drm.sdk.customobject.Call

def message = (Call.parameters.message as String)

return message.reverse()
```

Here we have invoked our Async Custom Object "**reverseString**" from the "**waitForExecution**" entry point. First, we iterate while checking the status of the Async job until we see that it has a status of either either "SUCCESS" or "FAILED" (QUEUED and CANCELLED are the other possibilities). Once we know that the execution is done, we read back the resulting value and output it.

Async Callback

Callback.png

It is also possible to invoke a CO Asynchronously with a "Callback" which will be executed after the first CO is done executing. In this case, the same size restrictions apply as are present in the Token / Poll approach (65536 bytes), but the output of the Called Object must be a Map, and will be passed into the Callback CO as the Parameters Map.

The following code example shows the Callback approach of initiating an Async job and then using the output returned from the first called CO to invoke another CO afterwards.

callbackExecution

```
//name: waitForExecution
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.drm.sdk.customobject.Call
import com.axeda.services.v2.AsyncCustomObjectExecutionStatus
import com.axeda.services.v2.ExecutionResult

def message = "This is our message"

// call our string reverse co
ExecutionResult er = Bridges.customObjectBridge.initiateAsyncWithCallback("reverseString", [message: message], "logOutput", 100)

// find the uuid for the Async execution, we will use this to look up the result
return er.succeeded.get(0).id
```

reverseString

```
import com.axeda.drm.sdk.customobject.Call

def message = (Call.parameters.message as String)

return message.reverse()
```

logOutput

```
import com.axeda.drm.sdk.customobject.Call

def message = (Call.parameters.message as String)

// must return a map for Callbacks
logger.info "Inside our Callback with value: $message"
```

Here we have invoked our Async Custom Object "reverseString" from the "callbackExecution" entry

point. Rather than wait for the execution to complete as in the Token / Poll approach, we instead take the results of the "reverseString" call and pass them into the "logOutput" custom object.

Todo: how do ER/CO timeouts work? With Callbacks?

Extended Data API

Tuesday, February 23, 2016 9:31 AM

extendedDataOverview.png

When the standard Asset-centric Domain Objects within the Platform are not sufficient to correctly model your solution, the Extended Data API provides a mechanism to extend the Platform to meet your Domain. This process of "Extended Data Modeling" allows a solution to store Objects which fit better than the standard Data Item / Alarm / Event Objects.

The Axeda Platform provides a variety of APIs to help with this Extended Data Modeling process. The APIs are collectively referred to as the "Extended Data API".

The Extended Data API runs from unstructured to highly structured constructs, each of which will be provided as a Getting Started example.

Data Accumulator - A bucket of Bytes which can be written in and streamed back out
Extended List - A named collection of Strings
Extended Map - A named collection of Name/Value pairs
Extended Objects - Structured, schema-defined Objects with flexible searching options

Data Accumulator

At the "least-structured" end of the Extended Data spectrum, we have the Data Accumulator. The Data Accumulator is bound to specific Assets in the system (as of 6.6, this will no longer be true). Once you have constructed a Data Accumulator for a specific Asset, you can then write "chunks" or Strings to the Accumulator. This is possible via Custom Objects or via an Expression Rule using the "accumulateData" Action.

The Data Accumulator is backed by a file, so there is no size restriction on the accumulator as a whole. There is, however, a limit on the amount of data that can be read out as a String (1k). Reading it out as a Stream, however, will have no size restrictions.

In this example, we create an Accumulator, write some bytes to it, and read them back again.

funWithAccumulators

```
import com.axeda.platform.sdk.v1.services.ServiceFactory
import com.axeda.platform.sdk.v1.services.data.DataAccumulatorService
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.services.v2.Asset
import org.apache.commons.lang.exception.ExceptionUtils

StringBuilder out = new StringBuilder();
out.append("Howdy!")

// Accumulations are scoped by device pre-6.6.
Asset myAsset = Bridges.assetBridge.find("TestModel||TestEdgeDevice")

// Get deviceId from a valid device.
Long deviceId = (myAsset.getSystemId() as Long);
```

```
// Get instance of service
ServiceFactory factory = new ServiceFactory();

DataAccumulatorService service = factory.getDataAccumulatorService();
// Typically, this value will be a primary key for your data.
String dataStoreIdentifier = "TestDataStore";

// Write some data. The first call to writeChunk will create internal data structure in the system behind the scenes.
service.writeChunk(dataStoreIdentifier, deviceId, "I'm writing a chunk!");
service.writeChunk(dataStoreIdentifier, deviceId, "A very exciting chunk!");
service.writeChunk(dataStoreIdentifier, deviceId, "Yet another Chunk!");

// Stream to StringBuilder. (for demonstration purposes only, do not stream large stores to memory)
InputStream is = service.streamAccumulation(dataStoreIdentifier, deviceId);

// Stream to StringBuilder.
byte[] b = new byte[4096];
for (int n; (n = is.read(b)) != -1;)
{
   out.append(new String(b, 0, n));
}

// Cleanup!
service.deleteAccumulation(dataStoreIdentifier, deviceId);
```

Here we have created our Accumulator, named "TestDataStore". This required us to first locate an Asset to which it will be attached. We found this Asset by using the ID format of "modelNumber||serialNumber" and passing it into the AssetBridge.

In this example we stream out the Accumulation into a StringBuilder. Is is possible (and useful!) to tie this stream to another stream, such as the output stream of a Scripto-accessed Custom Object.

Extended Lists

The Extended List provides a convenient way for storing lists of Strings in a persisted way on the Platform. A List is essentially a list of String values (up to 256 characters each) which is associated with a "name" (up to 128 characters). Each List can contain up to 1,000 entries. It is important to note that the List actually behaves like a Set, in that it cannot have duplicate entries.

In the following example we create an ExtendedList implicitly, then look it back up by name, and by contents.

funWithLists

```
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.services.v2.ExtendedList
import com.axeda.services.v2.ExtendedListCriteria
import com.axeda.services.v2.FindExtendedListResult

// we can simply append to a list, and if it does not exist it will be created on the fly
Bridges.extendedListBridge.append("TestList","FirstValue")
```

```
// look up our list based on the name
ExtendedList myList = Bridges.extendedListBridge.find("TestList")
logger.info "Found myList with a size of: ${myList?.list?.size()}"

// now find it by the value
ExtendedListCriteria criteria = new ExtendedListCriteria([entry:"FirstValue"])
FindExtendedListResult result =  Bridges.extendedListBridge.find(criteria)
logger.info "Found the test list by the value. # of Lists returned: ${result.lists.size()}"
```

Extended Map

Extended Maps are the sister construct to the Extended List, and can contain up to 1000 entries with 128-character key / 256-character value pairs. These Extended Maps can be constructed implicitly by appending values, which is the most typical use case for adding rows. Like all maps, there can be no duplicate keys in the map, so appending a pre-existing key has the effect of changing the value.

The example provided below does a simple create by appending a new key/value pair to a non-existent map. After this, the map is read back by name, and search for by one of the values. It is also possible to search based on keys, providing a flexible way to locate maps based on their contents.

funWithMaps

```
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.services.v2.ExtendedMapCriteria
import com.axeda.services.v2.FindExtendedMapResult

Bridges.extendedMapBridge.append("TestMap","TestKey","TestValue")

logger.info "Found Map of size: "+Bridges.extendedMapBridge.find("TestMap").map.size()

ExtendedMapCriteria criteria = new ExtendedMapCriteria([key:"TestKey"])
FindExtendedMapResult result = Bridges.extendedMapBridge.find(criteria)

logger.info "Found map by value of size ${result.maps.size()}"
```

Extended Objects

Extended Objects present one of the most complex and flexible features in the Axeda Platform, and certainly are the most complex component of the Extended Data API.

The best way to look at Extended Objects is to view them as traditional database-backed Objects in any Object Oriented language. First you have to decide the Object's properties or "interface", then you create "instances" of that Object. These concepts translate to the "ExtendedObjectType" and the "ExtendedObject" in the Platform.

Thie ExtendedObjectType will specify the names and data types of the properties available within Extended Objects of that type. The legal property types are: "String","Integer","Double","Date", and "Boolean". These property types help define what search capabilities will be available when searching for Extended Objects of this Type.

Given the complexity of these implementations, the actual ExtendedObjectType and ExtendedObject functionality has been embedded in utility methods, which are provided. These can be reviewed for more detail / flexibility, but they are known to work as-is, and will save you a lot of coding time.

createExtendedObjectType

```
import com.axeda.platform.sdk.v1.services.ServiceFactory
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObjectService
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObjectType
import com.axeda.platform.sdk.v1.services.extobject.PropertyDataType
import com.axeda.platform.sdk.v1.services.extobject.PropertyType

// obtain a handle to the Service
ExtendedObjectService eoSvc = new ServiceFactory().extendedObjectService

// build up a Map of legal properties
// these are going to be the strongly-types fields available to us within a given Extended Object
// each property has to have a specific type, one of the following: String,Integer,Double,Date,Boolean
def properties = new HashMap<String,PropertyDataType>()
properties.testString = PropertyDataType.String
properties.testInetegr = PropertyDataType.Integer
properties.testBoolean = PropertyDataType.Boolean
properties.testDate = PropertyDataType.Date
properties.testDouble = PropertyDataType.Double


ExtendedObjectType eoType = createExtendedObjectType(eoSvc,"our.test.class.name","Test Extended
Object","This is the first Extended Object Type we have created",properties)

return ["Content-Type":"text/plain","Content":eoType?.id]


/**
 * Creates a fully qualified ExtendedObjectType and persists it to the database.
 *
 * @note if the ExtendedObjectType already exists then an Exception is thrown and should be caught by
the caller.
 *
 * @param eoSvc - the Extended Object Service
 * @param className - the class name for the new Extended Object Type.
 * @param displayName - the display name for the new Extended Object Type.
 * @param description - the description of the new Extended Object Type.
 * @param propertyTypes - A Map of property types that should be attached to the new Extended
Object Type.
 *
 * @return The newly created ExtendedObjectType from the database.
 */
private ExtendedObjectType createExtendedObjectType(
      ExtendedObjectService eoSvc,
      String className, String displayName, String description, Map<String, PropertyDataType>
propertyTypes) {
   def result

   if (!eoSvc.findExtendedObjectTypeByClassname(className)) {
      def eoType = new ExtendedObjectType()
      eoType.className = className
```

```
      eoType.displayName = displayName
      eoType.description = description

      propertyTypes.each { name, dataType ->
         def propertyType = new PropertyType()
         propertyType.name = name
         propertyType.dataType = dataType
         eoType.addPropertyType(propertyType)
      }

      // ordinarily we could just return this thing after the createExtendedObjectType call, but you see
there is a
      // bug where the creation works but does not attach the propertyTypes that go along with it on the
      // return extended object. so instead we need to manually retrieve the objecttype.
      eoSvc.createExtendedObjectType(eoType)

      return eoSvc.findExtendedObjectTypeByClassname(className)
   }
   else {
      throw new Exception("ExtendedObjectType: '${className}' exists already.")
   }
}
```

createExtendedObject

```
import com.axeda.platform.sdk.v1.services.ServiceFactory
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObject
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObjectService
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObjectType
import com.axeda.platform.sdk.v1.services.extobject.Property
import org.apache.commons.lang.exception.ExceptionUtils
import java.text.SimpleDateFormat

// obtain a handle to the Service
   ExtendedObjectService eoSvc = new ServiceFactory().extendedObjectService

// we first need to know what type of ExtendedObject we are creating
   ExtendedObjectType eoType = eoSvc.findExtendedObjectTypeByClassname("our.test.class.name")

   // then we decide what property values to put in it
   // Date will not be parseable without it being in a known format, we recommend:
"yyyy'-'MM'-'dd'T'HH:mm:ss'Z'"

   String dateFormatString = "yyyy'-'MM'-'dd'T'HH:mm:ss'Z'"
   SimpleDateFormat dateFormat = new SimpleDateFormat(dateFormatString);
   dateFormat.setTimeZone(TimeZone.getTimeZone("UTC"));
   def dateString = dateFormat.format(new java.util.Date() - 1)

   // referenceID is an int, usually used to tie this to an internal object via the object via the internal
object's ID
   def properties = [:]
   properties.testString = "Any old String value in here"
   properties.testInetegr = "42"
   properties.testBoolean = "true"
```

```
   properties.testDate = dateString
   properties.testDouble = "3.141512"

// now use our helper method to create the content
   ExtendedObject exObj = createExtendedObject(eoSvc,eoType,-1,"MyFirstExObj",properties)

   return ["Content-Type":"text/plain","Content":exObj?.id]


/**
 * Creates a fully qualified ExtendedObject and persists it to the database.
 *
 * @param eoSvc - The Extended Object Service
 * @param objectTypeOrObjectTypeClassName - the object type or class name of the new Extended
Object.
 * @param referenceId - the internal object id. can point to an axeda domain object or whatever.
 * @param displayName - the display name (extended/external key) of the new Extended Object.
 * @param properties - A map of property names and values that should be added to the Extended
Object.
 *
 * @return The newly created ExtendedObject from the database.
 */
private ExtendedObject createExtendedObject(
      ExtendedObjectService eoSvc,
      objectTypeOrObjectTypeClassName, def referenceId, def displayName, Map<String,String>
properties) {

   if (!referenceId || referenceId < 1) { referenceId = 1 }

   ExtendedObjectType eoType = null
   if (objectTypeOrObjectTypeClassName instanceof String) {
      eoType = eoSvc.findExtendedObjectTypeByClassname(objectTypeOrObjectTypeClassName)
   }
   else {
      eoType = objectTypeOrObjectTypeClassName
   }

   def eo = new ExtendedObject()
   eo.extendedObjectType = eoType
   eo.externalClientKey = displayName
   eo.internalObjectId = referenceId

   properties.each { String name, value ->
      def property = new Property()
      property.propertyType = eoType.getPropertyTypeByName(name)
      property.value = value
      eo.addProperty(property)
   }


   // result.
   eoSvc.createExtendedObject(eo)
}
```

findExtendedObject

```
import com.axeda.platform.sdk.v1.services.ServiceFactory
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObject
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObjectSearchCriteria
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObjectService
import com.axeda.platform.sdk.v1.services.extobject.ExtendedObjectType
import com.axeda.platform.sdk.v1.services.extobject.Property
import com.axeda.platform.sdk.v1.services.extobject.PropertySearchCriteria
import com.axeda.platform.sdk.v1.services.extobject.expression.PropertyExpression
import com.axeda.platform.sdk.v1.services.extobject.expression.PropertyExpressionFactory
import org.apache.commons.lang.exception.ExceptionUtils
import java.text.SimpleDateFormat

   // obtain a handle to the Service
   ExtendedObjectService eoSvc = new ServiceFactory().extendedObjectService

   // construct the criteria for searching
   // this is made up of the outer criteria, which includes a propertySearchCriteria
   // within this propertySearchCriteria we embed a PropertyExpression, which can be
   ExtendedObjectSearchCriteria criteria = new ExtendedObjectSearchCriteria( [
         extendedObjectClassName : "our.test.class.name",
         propertySearchCriteria : new PropertySearchCriteria([propertyExpression :
PropertyExpressionFactory.like("testString","Any old%")])
   ])

   List<ExtendedObject> exObjs = eoSvc.findExtendedObjects(criteria,-1,-1,null)


   return ["Content-Type":"text/plain","Content":exObjs?.size()]
```

When it comes to finding ExtendedObjects, a very robust and flexible criteria structure is provided. Most of this flexibility surrounds the ability to craft queries based on Property values within ExtendedObjects. Properties can be compared against value using the following methods in the PropertyExpressionFactory class: eq(),ne(),like(),gt(),lt(),gteq(),lteq(). In this example we use the "like" method. NOTE: it is also possible to chain multiple PropertyExpressions together by using the and(),or() and not() methods.
It is common to only search within a given ExtendedObjectType to guarantee that there are no unexpected matched in the result set. We limit this in the ExtendedObjectSearchCriteria in this example.

Using the V2 Services

Tuesday, February 23, 2016 9:34 AM

Using the V2 Services

Object Structure - References

There are many objects in the SDK which are too heavy to serialize every time one interacts with the SDK. These objects have "Reference" objects which are simple objects that have just an id field and a systemId field. These object are used as place holders – or "references" – for the 'larger' object.

You must fill in the proper value, be it the platform identifier (systemId) or the unique field which identifies the object (the id field).

If both systemId and id are supplied, the Platform will use the systemId.

Identifiers

All objects effectively have two different ways to uniquely identify any given object type. The two different types of identifiers are the system id (currently this looks like a numerical long value, but that is not guaranteed). The other identifier is one or more fields in the object which uniquely identify a given object type by using the values of these fields, and can be thought of as the "human" id. You can provide either the actual system id or the identifying values (in the non-reference type) by filling out the proper id field.

Multiple Fields Composing the "Human" Id

In some cases – such as Asset – more than one field represents the object. In the case of Asset there are two such fields: modelNumber and serialNumber. In these cases a predefined delimiter has been assigned to represent the logical split of the two fields. This concatenation string is currently two pipe symbols: ||. Continuing with the Asset example, if the Asset had modelNumber of "myModel" and a serialNumber of "theSerialNumber," the id for this object would be: myModel||theSerialNumber.

Note that the use of a concatenation string precludes the use of this string in ids. This means that should you know your object will contain the given concatenation string, you should use the Find method, which takes a criteria, or you should use the findById method, which takes a systemId.

Invoking the SDK

Currently the four basic CRUD (Create, Read, Update, Delete) methods are available to each object listed in this document. For any operation the same pattern will be followed for every invocation. The general pattern is:

```
{operationName}.{objectName - camelCased}({instance of type objectType}.
```

Every object type has two read methods named 'find' that follow the same basic pattern listed above, as well as a findById method. The two different find operations are find (using the "human" id) and find (by criteria).

The find operation will return either the object being searched for, or it will return a null.

The findById operation takes a systemId. If the object is not found, or if the user does not have the proper privileges, it will return null.

The find by criteria operation takes a criteria object which will be named {ObjectType - Upper Camel Case} Criteria, and it will return a Find{ObjectType - UpperCamelCase}Result. These result objects will always be paginated, and will return a collection of the object type being searched for.

Bridges

All of these methods are also available by obtaining a reference to the actual 'bridge' in question. All bridges follow the {objectName}Bridge pattern, and can be obtained from the implicitly injected 'bridges' object by issuing a get. For example, to get an AssetBridge one would do either:

```
com.axeda.sdk.v2.dsl.Bridges.getAssetBridge(); //static reference
```

or:

```
bridges.getAssetBridge();
//instance reference to an instance of com.axeda.sdk.v2.dsl.Bridges
```

or using a more 'Groovy' approach, simply:

```
bridges.assetBridge;
//instance reference to an instance of com.axeda.sdk.v2.dsl.Bridges but using Groovy //specific notation
```

Find "by human id" Example - based on the Asset object

```
import com.axeda.services.v2.*;
def asset = find.asset("{id}")
```

Find "by criteria" Example - based on the Asset object

```
import com.axeda.services.v2.*;
def assetCriteria = new AssetCriteria(serialNumber:"someSN")
def assetResult = find.asset(assetCriteria)
```

Find "by system id" Example - based on the Asset object

```
import com.axeda.services.v2.*;
def asset = find.assetById("{id}")
```

The three operation methods are create, update, and delete.

Create Example - based on the Asset object

```
import com.axeda.services.v2.*;

def assetToCreate = new Asset(model:new ModelReference(modelNumber:"some model"), serialNumber:"some SN")

... //set any other asset properties which are able to be set

create.asset(assetToCreate)
```

Update Example - based on the Asset object

```
import com.axeda.services.v2.*;

def assetToUpdate = find.asset("{id}")

... //set any asset properties which are able to be updated

update.asset(assetToUpdate)
```

Delete Example - based on the Asset object

```
def assetToDelete = find.asset("{id}")

delete.asset(assetToDelete)
```

Searching

All operations will return either a "FindResult" for searches, which can return more than one result, or a singular object of the type being looked up when searching by human or system id.

Find operations should never return an exception. Should you encounter an exception while performing a find operation, please contact Axeda support.

By Human Id

Searches by id will return a single result of the type being searched for. Simply provide the id of the object to the search. When the object is not found a null will be returned.

By System Id

Searches by id will return a single result of the type being searched for. Simply provide the id of the object to the search. When the object is not found an exception will be thrown.

By SearchCriteria

In order to find objects by other than the id, a "SearchCriteria" will need to be used and passed to the find

operation. The result will be a "FindResults" object which will contain the criteria originally used to search for the results, as well as a view of the results. The "SearchCriteria" contains paging information embedded within it to make it easy to page results. If paging information was not supplied originally, the default paging settings will be used, and these values will be set in the returned "SearchCriteria." The total count of objects is returned, along with any matching results up to the page size indicated.

Wildcards

Searching for items using a wildcard is supported by presenting an asterisk (*) as part of the value which one is searching for. This matrix describes how the wildcard search is expected to work:

Given a field X has 3 separate values (as in 3 rows in a database):

"foo","snafoo","foobar"
"foo" matches only :"foo"
"foo*" matches both: "foo","foobar"
"*foo*" matches all: "foo","snafoo","foobar"
"*foo" matches both: "foo","snafoo"

Creating/Updating/Deleting Objects

Operations which perform an action on one or more objects will always return an ExecutionResult, which will contain all information necessary to correlate successful operations and failures. If any exceptions are thrown and an ExecutionResult is not returned, this is a leaking of the SDK and should be filed as a bug.

Execution Result

The ExecutionResult class has three fields: an optional Integer field called totalCount which represents how many records were affected by the execution, a list of FailedOperation objects which represents the failures that have occurred during the execution, and a list of SuccessfulOperation objects which represents a correlation of given input and id.

Total Count

The total count field is optional because when a failure occurs which prevents any executions from actually being invoked (like a validation issue), it is inconsistent to set this value because an operation would have been performed successfully. If the operation succeeds the totalCount will be set.

Failed Operation

FailedOperation objects are returned when expected or unexpected issues are encountered. The class contains three fields: a reference to the object which caused the failure, a code, and a message. The code and message are required elements and will always be populated.

The message is an English only message meant to assist in debugging. Should you want to use the message, you are free to do so, however, at this time the message is not internationalized and will be in English.

If you would like to internationalize the error, the code and the target should be used. If the code is numeric, it is guaranteed to be unique per "error type." If there is a code and a target, these two fields combine to form a unique error.

The ref field will be used to refer back to what the object sent in was referencing. This field is optional, but will generally be populated with the id of the object that caused the error. In the case where no id is supplied,

the "human" id will be used (if possible). In some cases it is unclear as to what value should be put into the ref field, and as such any value would lead to possible confusion. To avoid this confusion, the field is optional.

Successful Operation

SuccessfulOperation objects are returned as the result from any execution which succeeds. The object will always contain two fields: a 'ref' and an 'id.' Both of these fields are optional, as it is not always required to set both when performing various operations. The fields will be set according to this table:

Operation	Ref	Id
Create	Set to the 'human' id for the object>	Set to the systemId of the object that was created.
Update	Set to whichever id was provided to update. If the id was provided, it will be the id. If the systemId is provided, the systemId will be set. If both the id and the systemId are provided, the systemId 'wins.'	The systemId of what was updated.
Delete	Set to whichever id was provided to delete. If the id was provided, it will be the id. If the systemId is provided, the systemId will be set. If both the id and the systemId are provided, the systemId 'wins.'	Not set.

Some ExecutionResults will contain SuccessfulOperation objects which are more narrowly defined than just an id and a ref. These objects will contain extra information returned, which will prevent the need for calling back to the system to 'find' that which was just created, or other relevant information. This information will come in the form of a different class extending the SuccessfulOperation. Casting the object to the more narrow version will give you the ability to access this extra information.

Extended Successul Operation Objects

Often times it might be advantageous to return extra information as the result of a save operation over and above the id of the object just created. This extra information, if not returned as part of the create operation, requires that you fetch the object after the create succeeds in order to gain access to these values. In these cases the SuccessfulOperation object will be extended in order to provide whatever useful information is relevant to the object just created.

Errors

As mentioned previously, any errors should be trapped and sent back to the client in the 'failures' structure. Some operations can operate on more than one object at a time. In these cases, if the 'main' object is updated, but 'child' object updates fail for some reason, these failures will be returned as errors in the failures structure.

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/using_the_v2_services.html>

The File Store

Tuesday, February 23, 2016 9:35 AM

fileStoreUpload.png

The Axeda Platform has the ability to manage files uploaded from an Axeda Agent, however until SDK v2 files were not easy to manage from the scripting engine. Now Axeda customers can benefit from the **FileInfo** domain object and access their files in a new and useful way.

A **FileInfo** Object represents the Metadata about a specific File in the FileStore, and it provides the pointer necessary to write / read the actual contents of the file. Most of your interactions with the FileStore will be through this class.

The **FileUploadSession** is used as a container for **FileInfos** when they are first being populated. The session provides a way for the container to map multiple **FileInfos** to the same upload process.

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/file_info.html>

Store File

- In this example we show how to use both the **FileInfo** and the **FileUploadSession** in conjunction to store files within the FileStore, and then retrieve them back out as streams. This storage process takes a short series of operations:

  1.) Create the **FileInfo** object and fill up the metadata associated with the file
  2.) Create a **FileUploadSession** object and associate the FileInfos
  3.) Save the **FileUploadSession**, then store the actual data for the files described by the **FileInfo**s

- To post a file to a script using FileInfo, use a cURL command

```
curl -k "http://yourdomain.axeda.com/services/v1/rest/Scripto/execute

/StoreFile?username=User&password=Pass" -X POST --data-binary "myFile.jpg"

 --header "Content-Type: image/jpeg"
```

Find FileInfo

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

FileInfoCriteria fiCrit = new FileInfoCriteria(
    filelabel : filename,
    tags : tag
   )
def findFileRes = fileInfoBridge.find(fiCrit)

def file = findFileRes.files?.getAt(0)
```

Create FileInfo

```
import com.axeda.services.v2.*
import static com.axeda.sdk.v2.dsl.Bridges

FileInfo myFile = new FileInfo(filelabel: filelabel,
                                                filename: filename,
                                                filesize: bytes?.size(),

                                                description: description,

                                                tags: tag
                                                )

FileUploadSession fus = new FileUploadSession(
Files: [myFile]
)
ExecutionResult fer = Bridges.fileUploadSessionBridge.create(fus);
myFile.sessionId = fer.succeeded.getAt(0)?.id
ExecutionResult fileInfoResult = Bridges.fileInfoBridge.create(myFile)

ExecutionResult er =
Bridges.fileInfoBridge.saveOrUpdate(fileInfoResult.succeeded.getAt(0).id,n
ew ByteArrayInputStream(bytes))
```

-

FileInfos must be created along with creation of a FileUploadSession. The exception being that if an existing FileInfo needs new file data uploaded, it must be associated or updated with an active session Id first. Create a new session, find and update an existing FileInfo with the new session Id and other metadata, then upload the new file data.

Security

FileInfo access security is primarily group level security, however, it is multi-faceted. The user that created the file and administrators always have full access to the file. Other users need to be associated with a group that is associated with aFileInfo that has been associated through one or more **FileInfoPrivileges** with either Read (R) or Update (U) rights. **FileInfoPrivileges**can be created either by group id or by group name. In addition, users must be associated with a group that has one of the following privileges:

view-filestore
upload-to-filestore
update-in-filestore
delete-in-filestore

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/file_info/storefile.html>

Download File

-

The downloading process is very straightforward once we have the ID of the actual File. This ID can be used to get the metadata by finding a **FileInfo** object through the**FileInfoBridge**, or it can be used (as we do in this case) to retrieve the contents of the file and stream them out directly.

-

Download File

```groovy
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*
import com.axeda.sdk.v2.exception.*

def contentType = parameters.type ?: 'image/jpg'
['Content':fileInfoBridge.getFileData(parameters.fileId), 'Content-Type':contentType]
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/file_info/downloadfile.html>

Enterprise Integration

Tuesday, February 23, 2016 9:37 AM

In addition to the numerous Developer Features and frameworks which have been described in the other Getting Started articles, the Axeda Platform provides a set of Developer Features which don't fit cleanly into any specific category. These features often arose from a specific integration need, and as such they typically deal with interaction between the Platform and external entities.

In this series of Getting Started articles we provide information on these addition Developer Features.

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/additionalFeatures.html>

External Credentials

- The External Credential framework was created to provide a secure, access-controlled repository for credentials that are used in integrations. A SFDC integration, for instance, will require several pieces of information to be included in all requests. The External Credentials framework provides a "Credential Lockbox" into which these credentials can be placed.

  All External Credentials access can be controlled via UserGroup access rights. This includes the ability to use (or "view") a Credential without being able to modify it. The combination of a Group-specific "view" privilege and the absence of the global "view-external-credential" privilege is the most common arrangement, in that it allows Users in the associated Groups to use the Credential as part of an integration Custom Object without ever knowing of the Credential's existence.

  In this example, we show the simple create and read of an External Credential.

- funWithCredentials

```
1  // create our empty credemntial
2  def credential = new ExternalCredential()
3
4  // add some basic information
5  credential.name = "MyCredentialName"
6  credential.apiKey = "API18274361827364"
7  credential.endpoint = "http://mentor.axeda.com"
8
9  // add a custom property. You get up to 10
10 credential.parameters.add(new NamedValue([name: "myCustomProperty",value:
```

```
11 "myPropValue"]))
12
13 // create the credential
14 externalCredentialBridge.create(credential)
15
16 // look it back up
17 FindExternalCredentialResult result = externalCredentialBridge.find(new
   ExternalCredentialCriteria(endpoint: "http://mentor.axeda.com" ))

   return result.externalCredentials?.get(0)
```

- Security

  Each Credential contains a userGroupsWithEditPermission and userGroupsWithViewPermission List. These can be populated with UserGroupReferences to provide access control to the Credential.
  All Credentials can always be seen by the User who first creates it. This is done to prevent a Credential from "disappearing" if the Credential groups are set wrong.

  From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/additionalFeatures/externalCredentials.html>

Web Resources

- The Web Resource framework is analogous to the WebReference framework provided in .NET. This feature allows developers to create a Platform-side client library for external SOAP WebServices. This client can then be accessed through the WebResourceBridge to retrieve the actual Proxy objects generated from the remote WSDLs. Each WebResource can include one or more WSDLs (and all imported child XSDs) in each library.

  The Web Resource is frequently used in conjunction with the External Credential feature to provide a transparent integration with an external service.

  The example provided here creates a local WebResource and provides an example of how it can be accessed. In the **createWebResource** file, we simply point to the target WSDL and create the WebResource library. We are returned all of the information we need to use this library, including:

  - The XML to add to our Maven settings.xml file

  - The example code for getting a handle to the proxy WebResource class

  If Maven is setup correctly, you will be able to see all of the available methods via code completion!

- createWebResource

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.sdk.v2.bridge.*
```

```
import com.axeda.services.v2.*

// create web resource
def resource = new WebResource()
resource.name = "Country Service"
resource.description = "Country Code free WebService"
resource.packageNamespace = "mentor.webresource"
resource.wsdlURLs.add("http://www.webservicex.net/country.asmx?WSDL")


ExecutionResult result = webResourceBridge.create(resource)
if (result.isSuccessful()) {
    // return the WebResource object serialized to ZML
    return
webResourceBridge.findById(result.getSucceeded().get(0).getId())
}
else {
    // if the create failed, just return the result - we want to know why
it failed
    return result;
}
```

useWebResource

```
import com.axeda.sdk.v2.bridge.WebResourceBridge
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.webresource.mentor.webresource.CountrySoap
import org.apache.commons.lang.exception.ExceptionUtils

try {
    WebResourceBridge bridge = Bridges.webResourceBridge
    // lookup the endpoint
    CountrySoap endpoint = bridge.getClientEndpoint("Country Service",
"country", "countrySoap12")

    // return the result of our query
    return ["Content-Type":
"text/plain","Content":endpoint.getCountryByCountryCode("US")]

} catch (Exception e) {
    return ["Content-Type":"text/plain","Content":
ExceptionUtils.getFullStackTrace(e)]
}
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/additionalFeatures/webResources.html>

Maven Configuration

- 

The Axeda Platform provides not only the ability to download WebResource client Jar files which have been created on the Platform, but starting in 6.6 your Axeda Platform instance will server as a Maven Repository for all of the libraries you need to access Axeda code from your IDE.

There are 2 files which need to be modified locally in order to enable the use of the Axeda Platform as a Maven Repository.

%HOME%/.m2/settings.xml

This file provides the list of User-scoped Maven environment variables. It may not exist by default, but can be created at any time simply by saving out a file of the correct format in this location. The full file is included below for reference. "%HOME%" refers to the User's home directory (i.e. C:\Users\jsmith).

In this file we specify the credentials for the Axeda Platform instance which we are telling Maven to use when downloading artifacts from the Axeda Platform.

pom.xml

At the core of every Maven project is the pom.xml file. This file specifies the complete makeup of the project, including dependencies and repositories from which to pull the dependency artifacts. It is for the latter purpose that we will be configuring our pom.xml.

NOTE: Artisan ships with an existing POM, and as of 6.6 will provide an archetype with the repository information properly configured.

The following examples are built upon the WebResource GettingStarted topic.

- settings.xml

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                      http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <!-- this is the typical configuration, pointing to the root folder for
your Maven repository. -->
  <localRepository>${user.home}/.m2/repository</localRepository>
  <interactiveMode/>
  <usePluginRegistry/>
  <offline/>
  <pluginGroups/>
  <servers>
    <server>
       <!-- this is the standard WebResource repo name, but any String
works so long as it matches the ID in the pom.xml -->
      <id>axeda.webresource</id>
      <!-- this user/pass combo is your standard Platform login -->
      <username>myusername</username>
```

```
      <password>mypassword</password>
    </server>
  </servers>
  <mirrors/>
  <proxies/>
  <profiles/>
  <activeProfiles/>
</settings>
```

pom.xml

```
<!-- this set of dependencies is sufficient for all of the Axeda,
WebResource and third-party libraries you will consume -->
    <repositories>
        <repository>
            <id>repository.jboss.org-public</id>
            <name>JBoss repository</name>
            <url>
https://repository.jboss.org/nexus/content/groups/public</url>
        </repository>
        <repository>
            <id>axeda.webresource</id>
            <url>
http://platform.axeda.com:80/services/v1/rest/webresource/repo</url>
        </repository>
    </repositories>

    <dependencies>
     <!-- this is the value returned from the WebResourceBridge.create()
call -->
        <dependency>
            <groupId>com.axeda.webresource.1</groupId>
            <artifactId>mentor-webresource</artifactId>
            <version>1.0.0</version>
        </dependency>
    <!-- the standard Axeda SDK library -->
        <dependency>
            <groupId>com.axeda</groupId>
            <artifactId>platform-sdk</artifactId>
            <version>6.5</version>
        </dependency>
    </dependencies>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/additionalFeatures/mavenSetup.html>

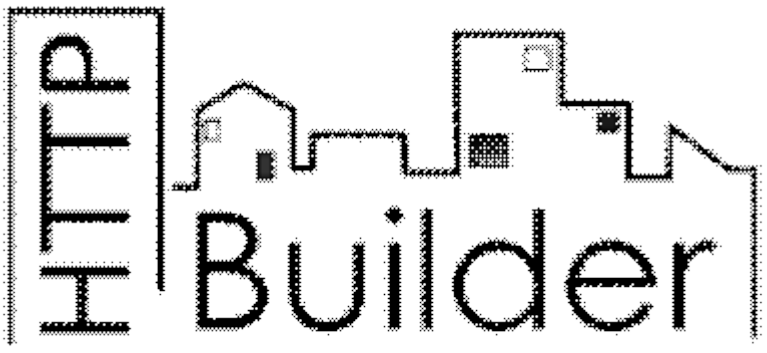
Third Party Libraries

Tuesday, February 23, 2016 9:39 AM

In addition to the APIs which are provided directly by Axeda, there are a number of 3rd party libraries which are available for import within Custom Objects. These can be accessed using the standard java-style import syntax.

In this series of Getting Started examples, we will show the imports and example calls from some of the most useful 3rd party APIs in the platform.

WebService Clients

- 

The HttpBuilder framework provided by the Groovy team provides a robust and flexible mechanism for making outbound HTTP calls from within Groovy code. We have included this library and its dependencies within the Platform to allow Custom Objects to take advantage of this flexibility.

For more details on the use of HttpBuilder, click here.

The following example demonstrates a simple GET request to the Google AJAX APIs to perform a search and output the results in JSON. This can be combined with the JsonSlurper to provide robust parsing and manipulation of returned JSON data.

After the HttpBuilder example, the exact same use case is shown using RESTClient, which is simpler to use, but slightly less flexible than HttpBuilder. Both are valid use cases to use out of the box (the Google API is not authenticated), so they serve as a great starting point for consuming external WebServices.

- funWithHttpBuilder

```groovy
import com.axeda.drm.sdk.scripto.Request
import groovyx.net.http.HTTPBuilder
import org.apache.commons.lang.exception.ExceptionUtils

import static groovyx.net.http.Method.*
import static groovyx.net.http.ContentType.*

try {

    int TENSECONDS  = 10*1000
    int THIRTYSECONDS = 30*1000


    def responseString = ""
    def http = new HTTPBuilder()

    // set the timeouts
    http.getClient().getParams().setParameter("http.connection.timeout",
new Integer(TENSECONDS))
    http.getClient().getParams().setParameter("http.socket.timeout", new
Integer(THIRTYSECONDS))


    http.request( 'http://ajax.googleapis.com', GET, TEXT ) { req ->
        uri.path = '/ajax/services/search/web'
        uri.query = [ v:'1.0', q: 'Axeda Corp.' ]
        headers.'User-Agent' = "Mozilla/5.0 Firefox/3.0.4"
        headers.Accept = 'application/json'

        response.success = { resp, reader ->
            assert resp.statusLine.statusCode == 200
            Request.logger.info "Got response: ${resp.statusLine}\n"
            Request.logger.info "Content-Type: ${resp.headers.'Content-
            responseString =reader.text
        }

        response.'404' = {
            responseString ='{Error: Not found}'
        }
    }

    return ["Content-Type":"application/json","Content":responseString]

} catch (Exception e) {
    return ["Content-Type":"text/plain","Content":
}
```

funWithRestClient

```
import groovyx.net.http.HttpResponseDecorator
import groovyx.net.http.RESTClient
import org.apache.commons.lang.exception.ExceptionUtils


try {

    def responseString = ""
    def client = new RESTClient('http://ajax.googleapis.com')
    HttpResponseDecorator response = client.get(
        path: '/ajax/services/search/web',
        query: [ v:'1.0', q: 'Axeda Corp.' ],
        contentType: "application/json"
    )
    if (response.success) {
        // we made it! digest the contents
        net.sf.json.JSONObject data = response.data
        responseString = data.toString(2)
    } else {
        // no megusta, failed request
        responseString = "{ message :
${response.getStatusLine().statusCode} -
    }

    return ["Content-Type":"application/json","Content":responseString]

} catch (Exception e) {
    return ["Content-Type":"text/plain","Content":
}
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/externalLibs/httpBuilderAndRestClient.html>

Apache Math

- 

There are some times when simple arithemtic just won't cut it!

For these tricky times we have exposed the Apache Common Math APIs to enable complex calculations. The following examples outline some of the basic operations available in this package.

More details are available directly from Apache here.

- funWithMath

```
package ThirdPartyLibs

```

```
3  import groovy.xml.MarkupBuilder
4  import org.apache.commons.lang.exception.ExceptionUtils
5  import org.apache.commons.math.stat.descriptive.DescriptiveStatistics
6
7  // initialize our XML response
8  def writer = new StringWriter()
9  def xml = new MarkupBuilder(writer)
10
11 try {
12
13     xml.Math {
14         // Get a DescriptiveStatistics instance
15         DescriptiveStatistics stats = new DescriptiveStatistics();
16
17         // Add the data from the array
18         (1..25000).each { stats.addValue it }
19
20         // Compute some statistics
21         double mean = stats.getMean();
22         double std = stats.getStandardDeviation();
23         double median = stats.getPercentile(50);
24
25         // return as XML
26         xml.Stats {
27             xml.Set("1..25000")
28             xml.Mean(mean)
29             xml.Std(std)
30             xml.Median(median)
31         }
32     }
33
34 } catch (Exception e) {
35     xml.error() {
36         faultcode ("Groovy Exception")
37         faultstring(ExceptionUtils.getFullStackTrace(e))
38     }
39 }
40
41 return ["Content-Type": "application/xml","Content":writer.toString()]
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/externalLibs/apacheCommonsMath.html>

SFTP Client

- 

For SFTP (not FTPS) support, we provide access to the Jsch library provided by JCraft. This provides Users with a flexible and proven library for running both SFTP and SSH sessions from within Custom Objects. This pairs perfectly with the External Credential feature to provide secure automated access to transmit or retrieve files.

Since there are no pre-existing open SFTP servers that are appropriate for use in this example, the code

provided is an example of usage, as opposed to a true cut & paste working example.

For details on what is available in the Jsch library, you can view the documentation here.

- SFTPClient

```
import com.jcraft.jsch.Session
import com.jcraft.jsch.ChannelSftp
import com.jcraft.jsch.JSch
import com.jcraft.jsch.Logger
import com.jcraft.jsch.UserInfo

com.jcraft.jsch.Session session = null
int port = 990


// get a handle to the main JSch instance
JSch sch = new JSch()

// add our logger in as the default JSch logger so that the debug output
ends up in our Groovy logs
JSch.setLogger(new Logger() {
    public boolean isEnabled(int i) {
        return true
    }

    public void log(int i, String s) {
        logger.info("Log(jsch," + i + "): " + s);
    }
});

session = sch.getSession("sshd", "hostname", port);

// create a new UserInfo Object to handle the authentication
session.setUserInfo(new UserInfo() {
    public String getPassphrase() {
        return "yourPassPhrase";
    }

    public String getPassword() {
        return "yourPassword";
    }

    public boolean promptPassword(String message) {
        return true;
    }

    public boolean promptPassphrase(String message) {
        return false;
    }

    public boolean promptYesNo(String message) {
        return true;
    }

    public void showMessage(String message) {
    }
});

```

```
54 // connect to the host
55 session.connect();
56
57 // obtain the contents of the file
58 String fileContents = readFile("/tmp/myFile.txt")
59
60 /
61 ************************************************************************
62 ***************
63  *
64  * Utility Methods below
65  *
66  ************************************************************************
67 ****************/
68
69 // utility method to read the contents of a file using sFTP
70 def readFile(String path, Session session) throws Exception {
71     ChannelSftp c = (ChannelSftp) session.openChannel("sftp");
72     c.connect()
73     ByteArrayOutputStream bos = new ByteArrayOutputStream()
74     InputStream is = c.get(path)
75     try {
76         byte[] buffer = new byte[256]
77         def count
78         while (-1 != (count = is.read(buffer))) {
79             bos.write(buffer, 0, count)
80         }
81     } finally {
82         is.close()
83     }

        c.disconnect()
        return new String(bos.toByteArray())
    }
```

-

Here we have first established a "session" with a remote host. To do this we had to manage the authentication component by providing a UserInfo object with the correct credentials. Once the session exists, a protocol can be selected which we will use over this session (in this case we choose sftp in line 67).

Here we assume that we know the name of the file we are looking to read: "/tmp/myFile.txt". It is also possible to list the files available, or "put" files in a remote folder.

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/externalLibs/sftpClient.html>

Oauth client

-

To provide easy access to Oauth protected services outside of the Axeda Platform, we have embedded the Scribe Oauth library. Scribe is a GitHub project which contains numerous pre-built connectors for use with many of the major Social Media platforms, but it also exposes the underlying plumbing for interaction with any in-house OAuth impl.

Full Scribe documentation can be found here.

In the following example we provide a simple Twitter Oauth example. Similar to SFTP, this is not a cut & paste type example as we do not have the valid tokens included in the source.

- updateTwitterStatus

```
1  import groovy.xml.MarkupBuilder
2
3  // initialize our XML response
4  def writer = new StringWriter()
5  def xml = new MarkupBuilder(writer)
6
7  def baos = new ByteArrayOutputStream()
8
9  try {
10
11     def oauth_key = "<your oauth private key>"
12     def oauth_secret = "<your oauth secret>"
13     def customer_key = "<your app-specific key>"
14     def customer_secret = "<your app-specific secret>"
15
16     //setup the scribe client
17     org.scribe.oauth.OAuthService service = new
18                                             .provider(org.scribe.builder.a
19 pi.TwitterApi.class)
20                                             .apiKey(customer_key)
21                                             .apiSecret(customer_secret)
22                                             .build();
23
24     // create the scribe request to the Twitter API URL - 'update my
25 status' in this case
26     org.scribe.model.OAuthRequest request = new
27 org.scribe.model.OAuthRequest(org.scribe.model.Verb.POST,
28 "http://api.twitter.com/1/statuses/update.xml");
```

```
29
30      //add the status to the body you want to tweet
31      request.addBodyParameter("status", "Test status update!");
32
33      //re-generate the access token and use it to sign the request (the
34  final step before sending anything 'oauth secured)
35      org.scribe.model.Token accessToken= new
36  org.scribe.model.Token(oauth_key, oauth_secret);
37      service.signRequest(accessToken, request);
38
39      //actually invoke the twitter api
40      org.scribe.model.Response response = request.send();
41
42      xml.Response() {
43          xml.Code(response.code)
44          xml.Message(response.body)
45      }
46
47  } catch (Exception e) {
48      StringWriter logStringWriter = new StringWriter();
49      PrintWriter logPrintWriter = new PrintWriter(logStringWriter)
50      e.printStackTrace(logPrintWriter)
51      logger.error("Exception occurred in OAuthTweet.groovy:
52  ${logStringWriter.toString()}")
53
54      xml.error() {
55        faultcode ("Groovy Exception")
          faultstring(e.getMessage())
          debug(baos.toString())
      }

  }

  return ["Content-Type": "application/xml","Content":writer.toString()]
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/gettingStarted/externalLibs/scribeOauth.html>

High Level Overview Topics

Tuesday, February 23, 2016 10:29 AM

Alarms

Tuesday, February 23, 2016 10:29 AM

Alarms

Alarms can be used to detect and correct Asset fault conditions. An Alarm can indicate that an error has occurred, a switch must be reset, a temperature must be adjusted, or even that a machine must be shut down and repaired.

Alarm Definitions and Instances

Alarm Definitions

On the Axeda Platform, an Alarm Definition assigns a name and a set of properties for an Alarm. Alarm definitions are created and managed at the Model level. The Configuration application in the Axeda Console can be used to create Alarm definitions.

Alarm definitions serve as templates for Alarm instances.

Alarm Instances

An Alarm instance contains information about a fault condition on a specific Asset. In the Axeda Platform API, Alarm instances are represented by the Alarm object, and can be created and manipulated using the Configuration application in the Axeda Console, or with the AlarmBridge SDK object.

Alarm instances can be generated in the following ways:

- By an Asset - Assets running an Agent (and certain wireless tracking devices) can send Alarm instances to the Axeda Platform.

- By a rule - The execution of a rule on the Platform can also generate an Alarm instance; when a certain condition is met, an Alarm is generated.

- By Web Services or SDK operations - Axeda Web Services and SDK operations can also be used to generate Alarm instances.

Alarm instances can also be used as inputs for Expression Rules. For example, you can create an Expression Rule that takes appropriate action when triggered by Alarm severity.

Life Cycle of an Alarm

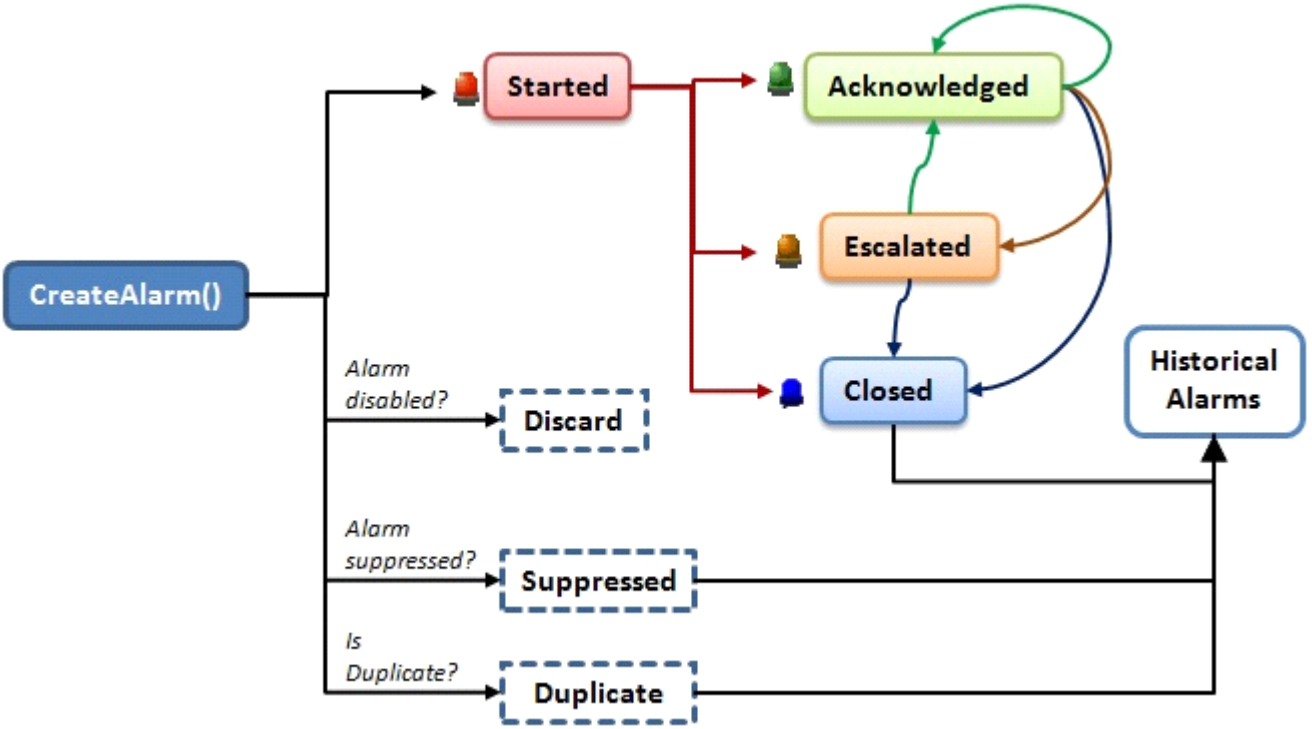
Service technicians can use the Asset dashboard in the Axeda Console to monitor and manage Alarm instances. Managing Alarm instances includes not only performing service actions to correct the problem, but also changing the state of the Alarm instance. Workflows for resolving problems with Assets can be adjusted to meet the requirements of different service organizations.

Alarm states can help show the progress of problem resolution. For example, the initial state of a new Alarm is Started. Once a service technician is ready to work on the problem, he or she can set the Alarm state to Acknowledged. If the problem requires additional assistance, the technician can set the Alarm state to Escalated and notify an engineer. The engineer can indicate that he or she is working on it by once again setting the Alarm state to Acknowledged. When the issue has been resolved, the Alarm state can then be set

to Closed.

If an Alarm is reported repeatedly, the disposition of the Alarm can be set to Suppressed. In the Suppressed state, any current instances of the Alarm remain, but future instances are closed and sent directly to the Historical Alarms page. When the problem has been resolved, the disposition of the Alarm can be reset to Unsuppressed, and new instances of the alarm will be set to Started. An Alarm can be disabled by setting the Alarm disposition to Disabled. In the Disabled state, all new instances of the Alarm are discarded and not recorded on the Platform.

The following diagram illustrates the life cycle of an Alarm.



As shown in the illustration, the life cycle of an Alarm has more steps than state changes. First, an Alarm instance is checked to see if it is a duplicate of an existing Alarm. The Platform checks only the name of the Alarm and the name of the associated Asset. If different Data Items or different values of a Data Item trigger the same Alarm, the Platform does NOT distinguish between the two instances of the alarm, each with a different Data Item or different value. It simply stores the information with the instance of the Alarm with which it was sent.

For example, a value of 100 for Temp_1 and a value of 200 for Temp_2 both generate the same Alarm. If a current Alarm instance exists with Temp_1 as the trigger, and the Asset sends in another instance of the Alarm with Temp_2 as the trigger, the Platform treats the second instance as a Duplicate. Temp_1 and its value are stored with the current Alarm, while Temp_2 and its value are stored with the duplicate instance of the Alarm. Similarly, if the Asset sends an alarm of the same name but a different severity, the Platform treats that Alarm as a Duplicate as well. Duplicate is a Terminal state, meaning that the Alarm instance is sent immediately to Historical Alarms and is not available to business rules. For a Duplicate Alarm instance, the state is set to Duplicate.

Once an Alarm instance is determined to not be a Duplicate, the Platform checks the initial disposition currently configured for the Alarm. The initial disposition for an Alarm determines how new instances of the Alarm will be processed. You have already seen how normal processing works - the Alarm is Started, and then its state can be changed as the problem is resolved. For an Alarm instance to be in the Started state, the Alarm definition must be Enabled and Unsuppressed. These settings are the defaults for any new Alarm

definition, whether defined through the Configuration application or created automatically when an instance of the Alarm is detected by the Platform.

Through the Axeda Configuration application and Axeda Console Web services, Alarm definitions can be disabled, re-enabled, suppressed, and unsuppressed for Models. Through the Axeda® Service application, Alarm definitions can be enabled, disabled, suppressed, and unsuppressed for individual Assets. These actions change the initial disposition and processing of a new Alarm instance, as follows:

- Disabled -- The Alarm definition has been disabled. A new Alarm instance will be discarded from the Platform immediately. No processing takes place, and the Alarm instance is not available for business rules.

- Suppressed -- the Alarm definition has been suppressed. A new Alarm instance is processed as follows: the Alarm instance is closed immediately upon receipt and sent directly to Historical Alarms. The state of the Alarm is set to Suppressed. The Alarm instance is not available for business rules.

Only if the initial disposition is Started is the Alarm state available for evaluation in an If statement in an Expression Rule, or for state changes through the SetAlarmState action. An Alarm with an initial disposition of Started can be transitioned through a SetAlarmState action, or through Web services (in addition to the Axeda® Service application), to one of the following states: Acknowledged, Escalated, or Closed, as shown in the illustration above.

The Alarm Object

The Alarm object contains the high-level properties that define a specific Alarm instance.

The AlarmBridge Object

An Alarm object can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

The AlarmBridge object provides Create, Read/Find, and Update operations for Alarm objects. You can also use the AlarmBridge to search the Alarm History, find AlarmState changes, and to set the value of the AlarmState.

Alarm Criteria

The AlarmBridge Find By Criteria method takes an AlarmCriteria object as input, and returns a FindAlarmResult object that lists (as a collection) the Alarms with the specified criteria. The FindAlarmResult object is always paginated.

Alarm search criteria can include any set of valid Alarm object properties. For a full description of AlarmCriteria properties and methods, see the Axeda API Specification.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/alarms.html>

Artisan Overview

Tuesday, February 23, 2016 10:29 AM

Artisan Overview
Developing with Axeda Artisan

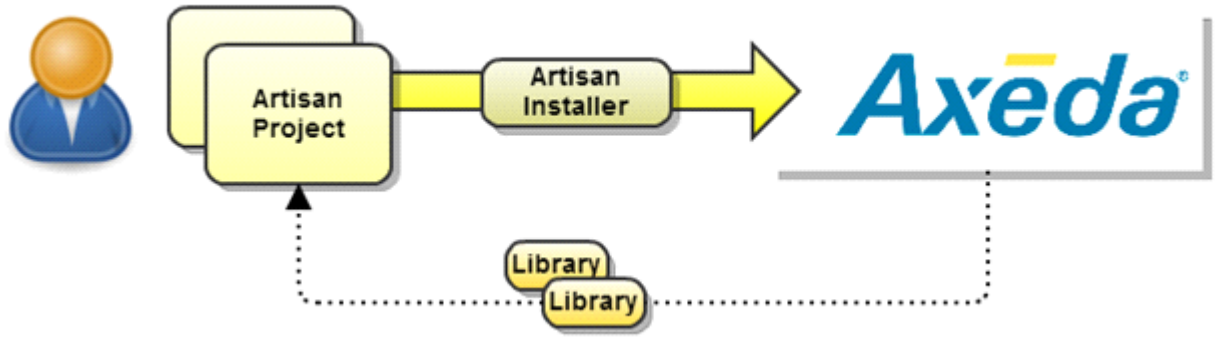
The Artisan framework provides a mechanism for developers to author, manage, and deploy applications on the Axeda Platform. Artisan is composed of three primary components:

Artisan Project – The collection of content and configuration files which represent an Application, and which are stored within a specific file / folder convention.

Artisan Installer – The application that uploads contents from a developer machine or build environment to the Axeda Platform.

Artisan Server – Each Platform instance acts as an Artisan Server. Developers point to this Server to obtain the libraries and example projects necessary to author and deploy Applications on the Platform.

These components provide an integrated experience for application development on the Axeda Platform.



Quick Start

Prerequisites

- A Java 7 installation
- A Maven 2+ installation
- Your favorite Java / Groovy IDE (Eclipse/IntelliJ)
- An account on an Axeda Platform instance, version 6.5 or higher
- An example Artisan project (more on this in other sections)

The following steps will get you up and running quickly with an Artisan project.

1. In the root folder of the Artisan project, configure the artisan.properties file with the user name, password and host name of your Axeda Platform instance.

2. From that folder, run the following command:

```
mvn package –f upload.xml
```

3. This will generate an artisan.log file, which will list the results of the installation.

4. Check your Axeda Platform instance to see that the application is installed.

Overview

A single Artisan Project is analogous to a single application running on the Platform. It is expected that there will be multiple Artisan Projects installed on any given Platform instance at any given time.

The Axeda Platform instance is responsible for providing all of the libraries which are needed for code completion during Custom Object authoring, as well as all Installer behaviors.

During the authoring process, it is common to install an application many times to a given sandbox instance. This cycle of installation is used to work out any bugs or issues found in development. Once the project is ready to be pushed to Production, the Artisan framework should be used to create a stand-alone installer which can be distributed to other teams for installation.

Artisan Project

Artisan projects use the Maven framework (http://maven.apache.org) for building, deploying and managing applications. Maven provides a convention for project structure, and is familiar to many Java developers.

NOTE: Both Java 7 and version 2+ of the Maven framework are required in order to use Artisan.

The Artisan Project format consists of numerous files and content folders. From the root of the project, the important files and folders are:

· **/artisan-starter-html** -- Under this folder, the “Custom Apps” composed of HTML, JavaScript, CSS, etc are stored. By default, the contents of this folder are grouped into a single custom application.
· **/artisan-starter-scripts** -- Under this folder all of the Groovy scripts which are being deployed to the Platform will be stored.
· **/metadata**: All of the configuration files related to this project will be stored under this folder. NOTE: This is optional, as described later.
· **artisan.properties** -- This is the primary configuration file for communicating to the Platform. Here you will setup the name of the Axeda Platform instance you are deploying to, and the credentials to use.
· **apc-metadata.xml** -- This is the primary file for defining the meta-data related to the Artisan project. This can contain all of the configuration for a given project, or may refer to other configuration files in the /metadata folder.

Artisan Installer

The Artisan Installer is a utility that is provided to install and uninstall content on an Axeda Platform instance. This can be run either in an “integrated” mode, which is how a developer will typically use it, or to generate a stand-alone installer which can be distributed and executed separately from the code.

All versions of the Installer create an artisan.log file that lists the results of the installation.

Integrated Mode

The Integrated Installer is typically run from the root folder of the Artisan Project. It is executed using the following maven command:

```
mvn package -P run-install
```

This will trigger the installer, which will upload the current contents of the Artisan Project to the configured server.

Stand-alone Mode

When running the installer in stand-alone mode, the first step is generating the installer from all of the content within your Artisan Project. This installer is a .jar file which contains all of the libraries and content required to perform an install or uninstall of the project.

The command for creating the installer is:

```
mvn package -Pmake-standalone
```

This will perform a series of operations resulting in a completed installer in the "/target" folder in the root of the project. NOTE: This will NOT result in the installation of the application – it will only result in the installer creation.

Installer Modes

Every Artisan project has a specific project name and version, which is used to determine the Artisan Installation mode.

This project information is found in the top-level tag of the **apc-metadata.xml** file. For example:

```
<apcmetadata version="1.2.0" project="Project1">
```

This project has the following properties:

ProjectName: Project1

Version: 1.2.0

These properties will be used to determine which mode the installer will be run in. The following installation modes are available:

Mode	Condition
Initial	No project of the same name has been installed previously.
Update	A project of the same name and version has been installed previously.
Upgrade	A project of the same name, but with a lower version number has been installed previously.
Downgrade	A project of the same name, but with a higher version number has been installed previously. NOTE: This will cause the installer to exit.
Uninstall	A project of the same name and version has been installed previously, and the Uninstall option has been supplied to the installer.

These install modes are used to control the initialization behaviors during the installation.

Installer Coverage

The Artisan Installer can deploy multiple types of content, all of which can be specified in the **apc-metadata.xml**. The following domain objects can be deployed:

	Example XML

Custom Object

```
<customobjects>
    <customobject>
      <name>YourScript</name>
      <type>Action</type>
      <sourcefile>Path/To/YoutrScript.groovy</sourcefile
      <params>
        <param name="yourparam" description="(REQUIRED) The description of the parameter"/>
      </params>
    </customobject>
</customobjects>
```

Model

```
<models>
    <model>
      <name>ModelName</name>
      <!--standalone or gateway-->
      <type>standalone</type>
      <DataItems>
        <DataItem>
          <name>DataItemName</name>
          <!--string or digital or analog-->
          <type>digital</type>
          <visible>false</visible>
```

```
            <stored>true</stored>
            <!--0 = no history,1 = Stored,2 = no storage,3 = on change-->
            <storageOption>1</storageOption>
            <readOnly>false</readOnly>
          </DataItem>
        </DataItems>
        <Properties>
          <Property>PropertyName</Property>
        </Properties>
      </model>
    </models>
```

Expression Rule

```
    <expressionRules>
      <rule>
        <name>Rule Name</name>
        <description>Rule Description</description>
        <enabled>true</enabled>
        <!—if applyToAll is set to true, any models will be ignored-->
        <applyToAll>false</applyToAll>
        <type>SystemTimer</type>
        <ifExpression><![CDATA[true]]></ifExpression>
        <thenExpression>
<![CDATA[ExecuteCustomObject("Some Custom Object")]]>
        </thenExpression>
        <elseExpression></elseExpression>
```

```
        <consecutive>true</consecutive>
        <models>
            <model>YourModelNumber</model>
        </models>
    </rule>
</expressionRules>
```

Rule Timer

```
<ruleTimers>
    <ruletimer>
        <name>RuleTimerName</name>
        <description>Rule Timer Description</description>
        <schedule>0 5 0/24 * * ?</schedule>
        <rules>
            <rule>Your Rule name</rule>
        </rules>
    </ruletimer>
</ruleTimers>
```

Custom Application

```
<applications>
    <application>
        <description>Application Description</description>
        <applicationId>Unique Application Name</applicationId>
        <indexFile>index.html</indexFile>
        <zipFile>artisan-starter.html/application.zip</zipFile>
        <!—optionally the application can be run in a tab-->
        <customTab>
```

```
<tabPrivilegeName>view-extended-tab-1</tabPrivilegeName>

<afterTab>YourTab</afterTab>

<showFooter>true</showFooter>

<tabNames>

  <label>

    <locale>en_US</locale>

    <name>YourTab</name>

  </label>

  <label>

    <locale>en_GB</locale>

    <name>Sir YourTab</name>

  </label>

  <label>

    <locale>ja_JP</locale>

    <name>YourTab-San</name>

  </label>

  <label>

    <locale>fr_FR</locale>

    <name>Le YourTab</name>

  </label>

  <label>

    <locale>es_ES</locale>

    <name>El Tabo</name>

  </label>

  <label>

    <locale>de_DE</locale>
```

```
                <name>Der Taben</name>
            </label>
        </tabNames>
    </customTab>
</applications>
```

Extended UI Module

```
<extendedUIModules>
    <extendedUIModule>
        <title> ModuleName</title>
        <height>800</height>
        <source>
            <type>CUSTOM_APPLICATION</type>
            <name>Your Custom App Name</name>
        </source>
    </extendedUIModule>
</extendedUIModules>
```

Initialization

```
<initialization>
    <step runOn="initial">
        <!-- body type can be either xml or json -->
        <script content-type="xml">YourScriptName</script>
        <startLabel>Text to display in the log before this step is run</startLabel>
        <endLabel>Text to display in the log after this step is run</endLabel>
        <body>
          <Request>
```

```
<Body>
  <Your>
    <XML>
      <Here/>
    </XML>
  </Your>
 </Request>
</body>
</step>
</initialization>
```

These examples can all be used within the **apc-metatada.xml** file directly. Each section (models, customobjects, etc.) can contain zero or more entries of the same type.

Importing Metadata Files

Although it is possible to include the entire project configuration in the **apc-metatada.xml** file, that can become unwieldy for large projects.

To support a cleaner, more streamlined metadata collection, it is possible to import other metadata files. These files need to be located in the "metadata" folder in the root of the project. They are clones of the **apc-metatada.xml** file, however, the project name and version elements will be ignored if they are included. It is permissible to create sub-folders under the metadata folder itself to provide more order.

The imports can be included within any of the sections of the **apc-metadata.xml** file as follows:

```
<import src="Path/To/YourFile.xml" />
```

NOTE: Only the section of the imported file that corresponds to the section in which the import was declared will be processed. For example, if you include an import in the <customobjects> section of the **apc-metadata.xml** file, only the <customobjects> section of the imported file will be executed.

Initialization

The process of initialization allows for Custom Objects to be invoked after the deployment of the project configuration is complete. This can be done in both installation and uninstallation modes.

The initialization process is broken into discreet steps, each of which is tied directly to an installation mode (Initial/Update/Upgrade/Uninstall). It is also possible to "clean up" a Custom Object that has been used in the initialization process. This can be done by including "deleteScriptAfter='true'" in the step.

For example, the following block:

```
<step runOn="initial" deleteScriptAfter="true">
```

```
  <script content-type="xml">RunInit</script>
  <startLabel>Starting Initialization</startLabel>
  <endLabel>Initialization Finished</endLabel>
  <body>
    <my>
      <xml/>
    </my>
  </body>
 </step>
```

Would do the following on the initial installation ONLY:

1. Call a Custom Object named "RunInit" using Scripto.
2. Write "Starting Initialization" to the log file.
3. Post the XML body "<my><xml/></my>" to this script.
4. Write "Initialization Finished", as well as the output from the RunInit script out to the log.
5. Delete the RunInit script.

See Also: Link to More Artisan Information

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/artisan_overview.html>

Assets

Tuesday, February 23, 2016 10:30 AM

Assets

An Asset is generally any piece of equipment that a company owns that could benefit by being remotely monitored and managed. Remote monitoring and management can reduce downtime, decrease service calls, and improve customer satisfaction. Assets can be any devices you would like to monitor with the Axeda Platform.

In order for the Axeda Platform to monitor an Asset, communication between the Axeda Enterprise Server and the Asset must exist. This communication does not need to be constant; the timing of the communication can be set up to suit the needs of your organization. Assets can communicate with the Axeda Platform over a variety of communication media, including wireless networks.

An Asset represents one instance of a Model, and requires a reference to a Model (the Model Name), as well as an individual serial number for the Asset.

Assets can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

Asset Properties and Methods

Every Asset must have a unique serial number, and must also be associated with a Model. If a name is not specified, the serial number will be used for the name value. Other available Asset properties include location and time zone, online/offline status, registration information, build version, device contact information, and whether or not further properties can be set in an associated AssetDetails object.

Asset object methods provide get and set operations for each of the Asset properties.

For a full description of Asset properties and methods, see the Axeda API Specification.

Asset Details

The AssetDetails object contains fields that require additional look-up in the database, and that therefore may affect performance. Because it is not required to include these fields with every Asset operation, they are split off into an AssetDetails object that can be queried only when necessary. AssetDetails fields include gateway parameters and references, asset groups, primary contacts, time zone, and token.

Asset Bridge

The AssetBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for Asset objects. You can also use the addDetails method to return an Asset with its details fields populated with data.

Asset Criteria

The AssetBridge Find By Criteria method takes an AssetCriteria object as input, and returns a FindAssetResult object that lists (as a collection) the Assets with the specified criteria. The FindAssetResult object is always paginated.

Asset search criteria can include any set of valid Asset or AssetDetails object properties. For a full description of AssetCriteria properties and methods, see the Axeda API Specification.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/assets.html>

Asynchronous Custom Objects

Tuesday, February 23, 2016 10:30 AM

Asynchronous Custom Object Execution

Overview

The Asynchronous Custom Object Execution feature is used to execute potentially long-running tasks in a separate thread, so that the thread requesting the asynchronous execution is immediately freed up. Optionally, you can also specify a "callback" Custom Object (i.e. post-processor) to be executed once the target Custom Object has completed execution. The callback Custom Object is also executed in a separate thread.

After the asynchronous execution request has been initiated, the callee may choose to "fire-and-forget," or can additionally:

- Periodically poll to obtain the status of the task
- Retrieve the results of the Custom Object Execution
- Cancel the Custom Object Execution
- Delete the results of the Custom Object Execution

Custom Object SDK

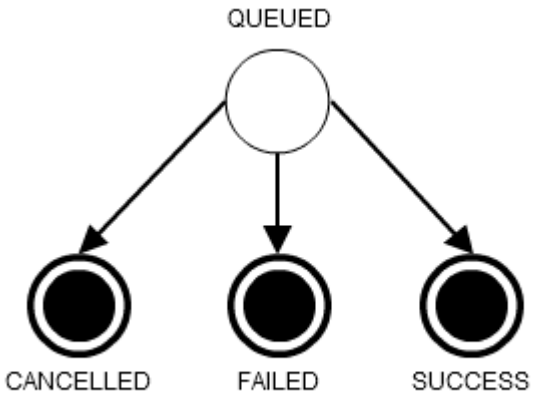
The asynchronous Custom Object execution feature is accessible via the V2 SDK. The following sections outline the API for asynchronous Custom Object execution.

Asynchronous Custom Object Execution States

Each "in flight" Asynchronous Custom Object Execution has an associated state, which can be one of the following:

- QUEUED (initial state)
- CANCELLED (end state)
- FAILED (end state)
- SUCCESS (end state)

The following diagram shows the valid state transitions:



SDK V2 Services

CustomObjectBridge

The CustomObjectBridge is an existing SDK class that provides functionality to execute and compile Custom Objects. This class has been extended to support initiation and management of asynchronous execution of Custom Objects.

Asynchronous CustomObjectBridge Methods

This section provides a brief overview of the CustomObjectBridge methods available for asynchronous execution of CustomObjects. For a full description of CustomObjectBridge methods, see the Axeda API Specification.

The following methods ONLY include the CustomObjectBridge methods that have been added to support asynchronous CustomObject execution:

Method	Description	Parameters	Return Value	Validation/Error Codes
initiateAsync	Initiates the asynchronous execution of a Custom Object.	customObjectName (mandatory) Map of parameters (optional) Timeout (optional)	UUID to uniquely identify the asynchronous execution	Validate that Custom Object exists. Validate that the size of serialized parameters are under max size (64 KB).
initiateAsyncWithCallBack	Initiates the execution of a CustomObject and the execution of a callback Custom Object after the target Custom Object execution is complete.	customObjectName (mandatory) callBackCustomObjectName (mandatory) Map of parameters (optional)	UUID to uniquely identify the asynchronous execution	Validate that the Custom Object exists. Validate that the size of serialized parameters are under max size

		Timeout (optional)		(64 KB). Validate that the Callback Custom Object exists.
getAsyncStatus	Retrieves the status of a previous request to initiate asynchronous execution of a Custom Object.	UUID, the unique identifier that was returned from a previous call to initiateAsync or initiateAsyncWithCallBack (mandatory)	An enumeration with 4 Possible Values: QUEUED CANCELED FAILED SUCCESS Will return null if the UUID parameter was not found in the database	Validate that the UUID exists in the database.
getAsyncResult	Retrieves the result of a previous request to initiate asynchronous execution of a Custom Object . Recommended usage is to only call this after confirming that the execution status is SUCCESS or FAILURE.	UUID, the unique Identifier that was returned from a previous call to initiateAsync or initiateAsyncWithCallBack (mandatory)	java.io.Serializable. This will either be the expected result, or an instance of Exception if an error occurred Will return null if the UUID parameter was not found in the database	Validate that the UUID exists in the database.
deleteAsyncResult	Deletes the persisted information about the asynchronous Custom Object execution keyed by the UUID, if and only if the status is SUCCESS or FAILURE or CANCELLED .	UUID, the unique identifier that was returned from a previous call to initiateAsync or initiateAsyncWithCallBack (mandatory)	N/A	Validate that the status is not QUEUED before deleting
deleteAsyncResult (ByCustomObjectName)	Deletes the persisted information about the asynchronous Custom Object execution keyed by the customObjectName, if and only if the status is SUCCESS or FAILURE or CANCELLED.	Custom Object Name (mandatory)	Number of Deleted Records	Validate that the CustomObject exists. Validate that the status is not is not QUEUED before deleting.

cancelAsync	Attempts to cancel the asynchronous Custom Object execution keyed by the UUID, provided that the status is QUEUED, otherwise it is a no-op.	UUID, the unique identifier that was returned from a previous call to executeAsync (mandatory)	N/A	Validate that the status is not QUEUED before the cancel attempt .
cancelAsync (ByCustomObjectName)	Attempts to cancel all of the asynchronous Custom Object executions keyed by the custom object name, provided that the status is QUEUED, otherwise it is a no-op.	Custom Object Name (mandatory)	Number of Cancelled Records	Validate that the CustomObject exists. Validate that the status is not QUEUED before the cancel attempt.

Timeouts for Custom Object Execution

When initiating an asynchronous Custom Object invocation, you can optionally specify a timeout value that the asynchronous invocation infrastructure will take into consideration when executing a Custom Object. The following are the maximum and minimum timeout values

- Minimum timeout value: 1 second
- Maximum timeout value: 600 seconds (10 minutes)

The SDK will accept any timeout value, but at execution time the timeout value may get overridden and reset to a reasonable value. Here is the pseudo-code for the overriding algorithm:

- if (timeout not specified) then timeout is MAX_TIMEOUT
- if (timeout < MIN_TIMEOUT) then timeout is MIN_TIMEOUT
- if (timeout > MAX_TIMEOUT then timeout is MAX_TIMEOUT

In addition, if a callback Custom Object was specified when initiating the process, the callback Custom Object execution will inherit the timeout value of the parent.

Custom Object Return Value Contract

Due to the asynchronisity of the CustomObject execution, the infrastructure must be able to store the results of the CustomObject execution in a persistent store. At a later time, the callee can retrieve and analyze these results in their business domain. As a result of this requirement, the return value of the CustomObject MUST be serializable based on the standard Java serialization specification. For more information, see the Java Object Serialization Specification
at: http://docs.oracle.com/javase/6/docs/platform/serialization/spec/serialTOC.html.

If after executing the CustomObject it is determined that the return value from the CustomObject execution is not serializable, the execution will be marked as FAILED, and any exception caught during the serialization

process will be stored for the callee to analyze.

Custom Object Return Value Contract When Using Callback

In addition to the contract for the CustomObject described in the previous section, if the callee also chooses to specify a callback CustomObject to be executed, there are additional contracts the callee must adhere to:

- The return value of the parent Custom Object must be of type java.util.Map<String,Serializable>
- The entire return value object graph Map<String,Serializable> must be serializable

If either of these contracts are not adhered to, the parent custom object execution will be marked as FAILED, any exception caught during the serialization process will be stored for the callee to analyze, and the callback will not be executed.

Example of a proper contract:

```
def inp1 = parameters.input1;
def inp2 = parameters.input2;
def out = \"out:\" + inp1 + \", \" +inp2;
def map = new java.util.HashMap<String,Serializable>();
map.put("result",out);
return map;
```

Example of an improper contract:

```
def inp1 = parameters.input1;
def inp2 = parameters.input2;
def out = \"out:\" + inp1 + \", \" +inp2;
def map = new java.util.HashMap<String,Serializable>();
map.put("result",out);
map.put("somethingnotSerializable",new java.lang.Thread());
return map;
```

Code Examples

The following sections provide examples of how the API is intended to be used for asynchronous execution of Custom Objects.

Initiate - Poll - Get Results - Delete Results

```
import com.axeda.drm.services.customobject.async.AsyncCustomObjectExecutionStatusEnum;
```

```
import com.axeda.services.v2.AsyncCustomObjectExecutionStatus;
import com.axeda.sdk.v2.bridge.CustomObjectBridge;
import com.axeda.sdk.v2.dsl.Bridges;
import com.axeda.services.v2.ExecutionResult;
import com.axeda.services.v2.SuccessfulOperation;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

// GET A REFERENCE to CustomObjectBridge

CustomObjectBridge customObjectBridge = Bridges.getCustomObjectBridge();

////////////////////////////////////////////
///  STEP 1: INITIATE ASYNC EXECUTION
////////////////////////////////////////////

String customObjectName = "yourCustomObjectNameToExecute";

// Optional: Set parameters to be passed in, can be null

Map<String,Serializable> parameters = new HashMap<String,Serializable>();
parameters.put("param1",1);
parameters.put("param2","somestring")

// Optional: Set timeoutInSeconds.  This will instruct the engine to timeout the execution
// of the Custom Object after X seconds
// Can be null, if null, then defaults to max value (600 seconds)

Integer timeoutInSeconds = new Integer(30);

// extract the UUID from the ExecutionResult

UUID uuid = null;
```

```
ExecutionResult executionResult = customObjectBridge.initiateAsync(customObjectName,
parameters,timeoutInSeconds);

if (executionResult.isSuccessful()) {

 SuccessfulOperation so = executionResult.getSucceeded().get(0);
 uuid = UUID.fromString(so.getId());

} else {

    return;

////////////////////////////////////////////////////////////////////////////
/// STEP 2: POLL UNTIL STATUS FOR THE EXECUTION IS SOMETHING OTHER THAN QUEUED
////////////////////////////////////////////////////////////////////////////

AsyncCustomObjectExecutionStatus status;
do {
   status = customObjectBridge.getAsyncStatus(uuid);
} while ( AsyncCustomObjectExecutionStatusEnum.QUEUED.equals(status) );

////////////////////////////////////////////////////////////////////////////
/// STEP 3: DO SOMETHING WITH THE RESULT.
//        THE RESULT IS THE CONTENTS OF WHATEVER YOUR CUSTOM OBJECT RETURNS
////////////////////////////////////////////

Serializable result = customObjectBridge.getAsyncResult(uuid);
if (AsyncCustomObjectExecutionStatus.SUCCESS.equals(status)) {
   // cast the result (Serializable) to the type expected
} else if (AsyncCustomObjectExecutionStatus.FAILED.equals(status)) {
   Exception theException = (Exception)result;
}

////////////////////////////////////////////
/// STEP 4: ONCE DONE WITH THE RESULT, DELETE IT
```

```
//////////////////////////////////////////

customObjectBridge.deleteAsyncResult(uuid);
```

Initiate with Callback - Delete Results By Custom Object Name

```
String customObjectName = "yourCustomObjectNameToExecute";
String callbackCustomObjectName = "yourCallbackCustomObjectName";

// Optional: Set parameters to be passed in, can be null
Map<String,Serializable> parameters = new HashMap<String,Serializable>();
parameters.put("param1",1);
parameters.put("param2","somestring");

// Optional: Set timeoutInSeconds.  This will instruct the engine to timeout the execution
// of the Custom Object after X seconds
// Can be null, if null, then defaults to max value (600 seconds)
Integer timeoutInSeconds = new Integer(30);

// extract the UUID from the ExecutionResult
for (int i= 0; i<10;i++) {
    customObjectBridge.initiateAsyncWithCallback(customObjectName,
parameters,callbackCustomObjectName,timeoutInSeconds);
}

//////////////////////////////////////////////////////////////////////////////////////////
/// STEP 2: ONCE DONE WITH THE RESULTS, DELETE ALL RESULTS FOR THE PARTICULAR CUSTOM OBJECT
//////////////////////////////////////////////////////////////////////////////////////////
customObjectBridge.deleteAsyncResult(customObjectName);
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/asynch_custom_objects.html>

Custom Objects

Tuesday, February 23, 2016 10:31 AM

Custom Objects

A Custom Object provides a mechanism for customizing rules and actions for use on the Axeda Platform. Based on the Groovy scripting language, the Custom Object feature provides a simple way to write custom Java code for the Axeda Platform.

Note: You can also use the ScriptoService to run scripts defined in the Axeda Platform. For more information about the ScriptoService, refer to the *Axeda® Platform Web Services Developer's Reference, v2REST*.

Custom Objects can be created and executed using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

Synchronous and Asynchronous Execution

Custom Objects can be executed both synchronously and asynchronously:

Synchronous execution — A Custom Object executes a Groovy script and waits for its output.

Asynchronous execution — A Custom Object executes a Groovy script, but does not wait for its output. This is sometimes referred to as “fire and forget” execution. Optionally, an asynchronous Custom Object execution can also periodically check the status of an initiated task, and retrieve the results of a task. It is also possible to register a “callback” Custom Object which will be invoked with the output of the first ansynchronously-invoked Custom Object.

For more details about asynchronous execution of Custom Objects, see Asynchronous Custom Object Execution.

Using Call and Request to Access Custom Object Parameters

The Call Object

The Call object holds the various information available when a Custom Object is invoked. A Groovy script can access this information by importing these static methods.

For example, if you did the following:

```
Bridges.customObjectBridge.execute("myscript", [foo: "bar"]);
```

Or, using the ScriptoService:

```
curl -X POST -d "foo=bar" -H "Content-Type: text/plain"
"http://host/services/v1/rest/Scripto/execute/myscript"
```

Then in a Groovy script, you could access the foo=bar parameters as follows:

```
import static com.axeda.drm.services.customobject.Call.*;
def fooParam = parameters.foo // should be "bar"
```

You could also access the current username:

```
def user = username // should be e.g., "admin"
```

For a full description of Call object methods, see the Axeda API Specification.

The Request Object

The Request object is an extenstion of the Call object that holds the various HTTP request information that is posted to Scripto.

The com.axeda.drm.sdk.scripto.Request object stores a specified subset of HTTP request information that is relayed to Scripto. The Scripto service extracts the information from a Scripto request and stores it in the Request object. When a particular Groovy script is executed, you can reference the static methods on the Request to access HTTP request information.

For example, if you did the following:

```
curl -X POST -d "foo=bar" -H "Content-Type: text/plain" "http://host/services/
v1/rest/Scripto/execute/MyScript"
```

Then in the Groovy code, you could extract the foo=bar parameters as follows:

```
import static com.axeda.drm.sdk.scripto.Request.*;
def fooParam = parameters.foo // should be "bar"
```

You could also extract the Content-Type header:

```
def contentType = Request.headers.'Content-Type' // should be "text/plain"
```

Note: For more information, refer to the API Specification.

Custom Object Permissions

Access control to Custom Objects is governed by privileges. Privileges can be applied globally to all Custom Objects, or to an individual Custom Object.

Individual Custom Object permissions can be added using the MappedPrivilege object. A Mapped Privilege represents an association between a privilege (permission) and a User Group. Each Custom Object can have a list of these Mapped Privileges.

The individual Privileges that can be applied to a Custom Object are Read, Write, and Execute. At the individual level, setting the Delete permission will have no affect. Instead, any user with the Write permission will be allowed to delete the Custom Object. By default, Custom Objects have no individual permissions applied.

Access to an individual Custom Object can be restricted based on permissions. If at least one User Group is assigned one of the Read/Write/Execute permissions for a specific Custom Object, access to that Custom Object will be restricted to that User Group (or any other User Groups that have been explicitly granted that permission for that Custom Object). Access is restricted based on specific permissions, so if only the Write permission is restricted, the access restriction will only apply to Write. If no User Group is assigned a particular permission, that type of access will not be restricted for the Custom Object, but access can still be restricted at the global level.

All users have global Execute permission by default. In order to perform Read and Write operations on a Custom Object, a user needs those global access privileges, or belong to a User Group with Read and Write permissions for the Custom Object. Access privileges for an individual Custom Object (assigned via User Groups) takes precedence over global access settings. For example, if Read access has been restricted for a Custom Object through a User Group, users with global Read permission will not be able to access that object unless they belong to that User Group.

The following table lists the Read, Write, and Execute (RWX) permissions for a Custom Object based on global and User Group permission settings. Note: in the scenario represented by the highlighted row, users who do not belong to the group should not have Write access, but currently they do because the global Write permission is enabled.

Global CO Read	Global CO Write	CO Read	CO Write	CO Execute	People Not in This Group	People In This Group	Notes
					X	X	only admin can update
			x		X	RWX	
		x			X	RX	only admin can update
		x	x		X	RWX	
	x				RWX	RWX	
	x		x		X	RWX	
	x	x			WX	RWX	
	x	x	x		X	RWX	
x					RX	RX	
x			x		RX	RWX	
x		x			RX	RX	only admin can update
x		x	x		RX	RWX	
x	x				RWX	RWX	
x	x		x		RX	RWX	
x	x	x			RWX	RWX	WRITE IS NOT RESTRICTED thus you get the read
x	x	x	x		X	RWX	
				x	-	X	only admin can update
			x	x	-	RWX	
		x		x	-	RX	only admin can update
		x	x	x	-	RWX	
	x			x	RW	RWX	
	x		x	x	-	RWX	
	x	x		x	-	RWX	
	x	x	x	x	-	RWX	
x				x	R	RX	only admin can update
x			x	x	R	RWX	
x		x		x	R	RX	only admin can update
x		x	x	x	R	RWX	
x	x			x	RW	RWX	
x	x		x	x	R	RWX	
x	x	x		x	RW	RWX	
x	x	x	x	x	-	RWX	

The CustomObjectBridge Object

The CustomObjectBridge object provides Create, Read/Find, Update, and Delete (CRUD) functions, as well as invocation operations, for Custom Objects.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/custom_objects.html>

Data Items

Tuesday, February 23, 2016 10:31 AM

Data Items

A Data Item is a named reading, such as a sensor reading or the available memory on a device.

On the Axeda Platform, a Data Item Definition assigns a name, a Model association, and a set of properties for an Data Item. Data Item definitions are created and managed at the Model level.

A Data Item Value holds the value of a Data Item on a specific Asset at a specific time. In the Axeda Platform API, Data Item Values are represented by the DataItemValue object.

Data Item Values -- such as hourly temperatures, odometer readings, or daily usage statistics -- are time-stamped in a sequence. Data Item Values can be used on the Axeda Platform in Expression Rules or Custom Objects that perform business logic.

The DataItem Object

The DataItem object contains the properties that define a Data Item.

The DataItemValue Object

The DataItemValue object contains the value of a Data Item on a specific Asset at a specific time.

DataItemValue Properties

- asset – The Asset reference.
- dataItem – The Data Item reference.
- timestamp – The time-stamp for the Data Item value.
- value – The Data Item value.

DataItemValue Methods

DataItemValue object methods provide Read (get) and Write (set) operations for each of the DataItemValue properties. For a full description of DataItemValue methods, see the Axeda API Specification.

The DataItemBridge Object

A DataItem object can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

The DataItemBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for DataItem objects. You can also use the DataItemBridge to get source and target Data Items.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/data_items.html>

Events

Tuesday, February 23, 2016 10:32 AM

Events

Events are very similar to Alarms in that they are sent up from an Asset to report a specific occurrence. The differ from Alarms, however, in that they do not have a "Lifecycle" associated with them.

Events simply "happen", and each and every occurrence of the Event will be logged by the Platform.

Event Structure

Name	The unique (on a per-Model basis) name of the Event
Description	A free-text description of the Event
Severity	An integer value indicating how severe the Event is
Date	The time when it happened

The EventBridge Object

An Event object can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

The EventBridge object provides Create, Read/Find, and Update operations for Event objects. You can also use the EventBridge to search the Event History.

Event Criteria

The EventBridge Find By Criteria method takes an EventCriteria object as input, and returns a FindEventResult object that lists (as a collection) the Event with the specified criteria. The FindEventResult object is always paginated.

Event search criteria can include any set of valid Event object properties. For a full description of EventCriteria properties and methods, see the Axeda API Specification.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/events.html>

Expression Rules

Tuesday, February 23, 2016 10:32 AM

Expression Rules

You can use Expression Rules to define your own business rules based on If-Then-Else expressions. Expression Rules provide a way to evaluate messages, data, and conditions, and then run actions based on the results of the evaluation. Expression rules must evaluate to a Boolean result (true or false, or 1 or 0) at runtime.

Alarm Trigger Example

You can create an expression rule triggered by an Alarm message that evaluates the severity of alarms reported by any one of the associated assets. If the alarm name is "alarmXYZ", and the severity for alarmXYZ is greater than 100, and the data item "machineFan" is not 1, then the Platform needs to run an action that writes a value to a data item on the related asset. You can also specify an Else portion that directs the Platform to write a different value to the data item. You would create the expressions and conditions for this rule as follows:

If: (Alarm.name == "alarmXYZ" && Alarm.severity>100) && (DataItem.machineFan.value != 1)

Then: SendDataItem(dataItemB, 1)

Else: SendDataItem(dataItemB, 0)

Format for an Expression Rule

An Expression Rule contains three expressions: If, Then, and Else. The Else expression is optional. The format for an expression rule is as follows:

If <expression> Then <expression> Else <expression>

The If Expression

The If expression defines system messages or information that you want the Platform to evaluate to true or false.

For example, the following expression determines if the object is less than 3 miles from the geofence "home":

If: DistanceToNamedGeofence("home", location) < 3

An If expression can be simple or complex. You can configure a rule that runs its actions every time alarm "AlarmXYZ" occurs for model "ModelABC" (simple) or only when alarm "AlarmXYZ" occurs with a severity greater than "100" and data item "Data123" has a value of "1" for model "ModelABC" (complex).

Note: For a MobileLocation trigger, you can specify an unqualified "location." For all other triggers, you need to qualify this as Location.location.

The Then Expression

The Then expression specifies the action(s) the Platform runs when the If expression evaluates to true. For example, the following expression writes the value 0 to the asset data item, "DataItemxyz":

Then: SendDataitem("DataItemxyz", 0)

The Else Expression

The Else expression specifies the action(s) the Platform runs when the If expression evaluates to false. For example, the following expression writes the value 1 to the asset data item, "DataItemxyz":

Else: SendDataitem(“DataItemxyz", 1)

Supported Actions, Functions, and Variables

The functions and variables that you can define for each <expression> vary based on the type of trigger selected, and whether you are defining an If, Then, or Else expression. For example, Alarm triggers support a variety of variables, including name, severity, time, and value. When you select an Alarm trigger, all functions and variables supported for that trigger are shown in the Expression Tree.

The ExpressionRule Object

The ExpressionRule object contains the properties that define the If-Then-Else expressions, as well as the Models and Assets associated with the Expression Rule. The ExpressionRule object can also contain information about the ExpressionRule author, creation and modification dates, the last modified user, and whether or not the ExpressionRule is a stand-alone object (versus being part of a state machine).

The ExpressionRuleBridge Object

Expression Rules can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

The ExpressionRuleBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for ExpressionRule objects. You can also use the ExpressionRuleBridge to validate expressions.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/expression_rules.html>

Files and Packages

Tuesday, February 23, 2016 10:32 AM

Files and Packages

Overview

The Axeda Platform provides file management capabilities that enable you to upload files, control access to files, assemble files into packages, and deploy packages to assets.

Working with Files

- In order to upload a file to the Axeda Platform, it must be defined in a FileInfo object.
- Access to files on the platform is controlled via user group associations set using the FileInfoPrivilege object.
- A FileUploadSession must be created to upload one or more files to the platform.

Deploying Software Packages

- A Software Package is a collection of information used to describe a package that includes instructions intended to be delivered to Assets for evaluation.
- Packages can be deployed to assets using Deployment objects.

Creating FileInfo Objects

In order to upload a file to the Axeda Platform, it must be defined in a FileInfo object.

In order to create a FileInfo object, a file name and file size (or MD5) are required. A number of optional fields enable you to set FileInfo properties to facilitate searching, package deployment, file locking, and access privileges.

Setting File Access Privileges

Access to files on the Axeda Platform – as defined in a FileInfo object – is controlled via user group associations set using the FileInfoPrivilege object. Group file access can be set using either GroupId or GroupName. The referenced group must have one of the following file access privileges:

- Files - View
- Files - Upload
- Files - Update
- Files – Delete

In addition, the user that created a file always has access to that file. Administrators always have full access to all files.

Uploading Files

FileUploadSession objects are sessions used to manage file uploads. Files can only be uploaded when associated with an active File Upload Session.

To create a File Upload Session, a user must either be an administrator, or be associated with a group that has the file upload privilege. When a File Upload Session is created, username and password are returned. These values can be stored so that the session can be accessed at a later time.

If an existing FileInfo object needs new file data uploaded, it must first be associated or updated with an active upload sessionId, as follows: create a new upload session, find and update an existing FileInfo object with the new session Id and other metadata, and then upload the new file data.

File Upload Sessions can be created and manipulated using the FileUploadSessionBridge object in the Axeda API. FileUploadSessionBridge methods can also be accessed via REST calls over HTTP.

Creating and Managing Packages

A Software Package is a collection of information used to describe a package that includes instructions intended to be delivered to Assets for evaluation.

Software Packages can be referenced by the internal identifier given to the object by the Axeda Platform, or they can be referenced using a user-generated and unique alternateId. Permissions can be applied to the package, thereby granting different groups access to the package. For example, a Read permission allows the package to be deployed, and a Delete permission allows the package to be deleted.

Until published, a Software Package can be edited. Once published, a Software Package can only be deployed or deleted.

Deploying Packages

A Deployment controls which Software Packages are delivered to one or more Assets. Without Deployments, no packages would be delivered.

Deployments can either be manual or automatic. The main differentiator is that manual Deployments are tied to a fixed set of Assets, whereas automatic Deployments will allow Assets – as they become available – to receive the specified package.

As with Software Packages, Deployments can have expiration dates, and can also have dependencies.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/files_and_packages.html>

Locations

Tuesday, February 23, 2016 10:33 AM

Mobile Locations

MobileLocations are a key component of any IoT solution. The Axeda Platform uses the MobileLocation of a Device to plot the Device on a Map, or to perform GeoLocation based Actions.

MobileLocation messages can be sent directly from a Device using eMessage, AMMP, or one of the supported Codecs. It is also possible to set the MobileLocation using Custom Objects.
The "Current" MobileLocation for a given Device will be set to the last reported MobileLocation, and the history of Locations which have been set can be queried.

Mobile Location Structure

Lat	The Latititude as a floating point number
Lng	The Longitude as a floating point number
Alt	A long value representing the altitude of the Device
Date	The time when the location was reported

Mobile Locations in Alarms

It is possible to report a MobileLocation along with an Alarm message. This is done by adding a MobileLocation object directly to the Alarm definition. This Alarm is then automatically "tagged" with the MobileLocation where it was reported, providing an accurate view of the location where Alarms are occurring.

GeoFences

MobileLocation proximity to specified locations or areas is defined via "GeoFences", which are named sets of coordinates which together describe an "Fenced-In" area.
There are 2 types of Geofences supported by the Platform:
- **Circle**: Created by specifying a center point (lat,long) and a radius
- **Polygon**: Created by specifying a series of points (lat,long) in order, causing a "fence" to be created from in between each pair of points.

Mobile Locations in Expression Rules

- In Geofence: Returns true if the Device is currently inside the named GeoFence
- Distance To GeoFence: Provides the distance from the Device to the nearest outer boundary of the GeoFence

These functions can be combined with other Expression Rules capabilities, such as Custom Object execution, to provide a robust MobileLocation solution.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/location.html>

Platform Settings

Tuesday, February 23, 2016 10:33 AM

Platform Settings

Axeda Platform administrators can configure Platform settings when installing the Axeda Platform. Administrators can also update Platform settings for an existing instance of the Platform.

Axeda administrators can access Platform settings through the Axeda API, or via a REST call over HTTP.

In the Axeda Platform API, configuration settings are stored in the PlatformSettings object. The contained Settings will have setting paths (Setting.path) relative to the Platform settings root path (PlatformSettings.rootPath). For consistency, all root paths are considered to reside under the /settings path. This will be handled transparently by the Platform by prepending the path if it is not explicitly put into the root path.

For example, using the root path:

/AxedaUI/rowsPerPage

will be treated by the Platform as the path:
/settings/AxedaUI/rowsPerPage.

Using the null, empty, or "/" root path will be treated by the Platform as the /settings root path. When the settings are obtained from the platform through find operations, the /settings prefix will always be present in the root path. Because the PlatformSettings object can be identified solely by the root path, it does not have an additional system id or alternate id, so these fields will never be populated. The detail and label fields will also not be populated.

Configuration Settings Layers

The configuration settings in the platform are structured in layers. The important idea is that each layer will be overlayed on the previous layer to form a complete view of the settings. Any setting defined in a lower layer will overwrite the same setting defined in an upper layer, if present. The three layers defined in the platform are (from top to bottom, with the Default layer being the highest layer, and the User layer being the lowest layer):

The Default Layer

This layer contains the default settings specified by Axeda, and these settings can have any path. These configurations can be read by the install layer, and they can be overwritten by the Installation and User layers.

The Installation Layer

This layer contains any settings set by the installation administrator. This layer can contain settings with any path under /settings. These settings will overwrite the settings already defined in the Default layer.
The InstallSettingsBridge is used to manage the settings in this layer.

The User Layer

This layer contains personalized settings for each user. Access to this layer is based on the currently logged in

user. The User layer cannot read or write to /settings/system path, as this path is reserved for the Default and Installation layers. Except for this reserved path, the User layer can contain settings with any other path under /settings. These settings will overwrite the settings defined in the Installation and Default layers. The UserSettingsBridge is used to manage the settings in this layer.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/platform_settings.html>

Remote Sessions

Tuesday, February 23, 2016 10:34 AM

Remote Sessions

Remote Sessions enable service personnel to access and communicate with devices hidden behind corporate firewalls, when those devices do not have a graphical desktop interface. Remote Sessions also allow for sharing of an Axeda Agent device's desktop. There are four types of remote sessions: Remote Terminal, Remote Application, Remote Browser, and Remote Desktop.

Remote Session Types

Remote Terminal Sessions

Remote Terminal sessions are connections from a local computer to a specific interface running on the Axeda Agent installed on an Asset. These connections are based on the standard protocols (Telnet and SSH). All communications to and from the Asset are shown in a Terminal Emulation applet window. When you type in standard Telnet commands, the applet window displays the responses from the Asset.

You can connect to Remote Terminal sessions in an exclusive manner, so that no one else can join the session, or in a non-exclusive manner, whereby other users with Remote Terminal privileges for that asset can join the session. If there are multiple users in a Remote Terminal session, you (the session initiator) have control of the session, and all other users have view-only privileges. You can "pass" control to another user, at which point you have view-only privileges until you take control again. Users can request and receive control. These control options are available in the Telnet applet window.

Remote Application Sessions

Remote Application sessions are connections from the local computer to a selected application running on the related Asset. The Axeda Console applications available for sessions must be defined in the project configuration for the Axeda Console Agent running on the Asset. The Remote Application tool uses Axeda Application Bridge technology to provide a virtual bridge between your local network and the network behind the Asset's firewall.

The Application Bridge is a Java applet (AppBridge.jar) that enables remote access sessions through the Platform, and it must be installed and running on your computer in order to start the remote session. When you start the Application Bridge (from the command line), you provide the remote access session ID (generated by the Axeda Platform when you request the session). The Application Bridge displays the number of the port (or a range of ports) defined for communication from the Asset to the Axeda Platform. You then provide this port number and the name of the computer hosting the Application Bridge (for example, "localhost" or an IP address), to connect to an application actively running on the target Asset.

Remote Browser Sessions

Remote Browser sessions are browser-based connections from the local computer to an Asset running an Axeda Agent and configured to support remote browser sessions. These Assets contain web browser-based user interfaces — for example, a printer containing a web server. All Remote Browser sessions are secure and do not expose the web server to the Internet.

The browser applications available for sessions must be defined in the project configuration for the Axeda Agent running on the asset. In addition, the Asset must be configured to support browser sessions. Assets configured to support browser sessions use a built-in web server to enable remote users to configure and

maintain the Asset. Because the Asset is hidden behind a firewall, there is no way to access it through a standard web browser.

Remote Desktop Sessions

Remote Desktop sessions enable you to share a remote Asset's screen or desktop; for example, to permit remote administration or operator education using that remote asset. Remote Desktop sessions can be started on a remote Asset with ("attended") or without ("unattended") operator intervention. Unattended sessions can be started on an Asset when it next communicates with the Platform.

The Quick Launch feature available through the Axeda Service application allows technicians to establish Remote Desktop sessions with an Asset, and automatically launch the appropriate viewer application for the Asset. For example, if the Asset is running Axeda Desktop Server, the Platform automatically launches Axeda Desktop Viewer, and also stores it locally on the computer that initiated the session. Quick Launch also updates the local copy of the viewer application whenever a new version is copied to the Platform.

The RemoteSession Object

The RemoteSession object contains the high-level properties that define a Remote Session.

The RemoteSessionBridge Object

A Remote Session can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

The RemoteSessionBridge object provides Create and Find operations for RemoteSession objects. You can also use the RemoteSessionBridge to start and end Remote Sessions, to find Remote Sessions, and to find the Remote Interfaces associated with a specified Asset.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/remote_sessions.html>

Users

Tuesday, February 23, 2016 10:34 AM

Users

In order to interact with the Axeda platform, a user must be defined in a User object. The User object contains information such as user name, password, and contact information. You can use the User object apiOnly property to restrict a user from accessing the Axeda Console UI. Setting apiOnly also prevents the user's password from expiring.

Access to Platform objects can be granted or restricted with User Groups. A User assigned to a User Group inherits all the access privileges associated with that User Group. For Users not assigned to the User Group, access is restricted.

The User Object

The User object contains the properties that define a specific User.

The UserBridge Object

A User object can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP.

The UserBridge object provides Create, Read/Find, and Update operations for User objects. You can also use the UserBridge to view and manage User passwords.

User Criteria

The UserBridge Find By Criteria method takes a UserCriteria object as input, and returns a FindUserResult object that lists (as a collection) the Users with the specified criteria. The FindUserResult object is always paginated.

User search criteria can include any set of valid User object properties. For a full description of UserCriteria properties and methods, see the Axeda API Specification.

From <https://mentor.axeda.com/magnoliaPublic/mentor/hlo/users.html>

Domain Objects

Tuesday, February 23, 2016 10:55 AM

AlarmBridge

Alarms can be used to detect and correct Asset fault conditions. An Alarm can indicate that an error has occurred, a switch must be reset, a temperature must be adjusted, or even that a machine must be shut down and repaired.

On the Axeda Platform, Alarm definitions are created and managed at the Model level, and serve as templates for Alarm instances. An Alarm instance contains information about a fault condition on a specific Alarm, and is represented by an Alarm object in the Axeda Platform API.

Alarms can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating Alarms through the API using the AlarmBridge object.

The AlarmBridge object provides Create, Read/Find, and Update operations for Alarm objects. You can also use the AlarmBridge to search the Alarm History, find AlarmState changes, and to set the value of the AlarmState.

The Delete operation is not supported for Alarm objects because an Alarm must pass through a state machine before it can be closed.

Methods

The following table provides a brief overview of the AlarmBridge methods, along with available REST calls. For a full description of AlarmBridge methods, see the Axeda API Specification.

Method	Description	REST Call
count	Counts the alarms that match the specified criteria.	POST <server>/services/v2/rest/alarm/count
create	Creates a new Alarm.	PUT <server>/services/v2/rest/alarm **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/alarm
delete **NOT SUPPORTED**	If invoked, throws an Unimplemented error.	DELETE <server>/services/v2/rest/alarm/id/{id}
find (by criteria)	Finds Alarms based on search criteria.	POST <server>/services/v2/rest/alarm/find
find (by IDs)	Finds Alarms based on a list of identifiers.	POST <server>/services/v2/rest/alarm/find

		ByIds
find (by alternate ID)	Finds an Alarm based on the alternate identifier.	GET <server>/services/v2/rest/alarm/id/{id}
findById	Finds an Alarm based on its platform identifier.	GET <server>/services/v2/rest/alarm/id/{id}
findOne	Returns the first Alarm found that meets specified search criteria.	POST <server>/services/v2/rest/alarm/findOne
findAlarmsInHistory	Searches the Alarm History and returns the Alarms that match the specified search criteria.	POST <server>/services/v2/rest/alarm/findInHistory
findAlarmStateChanges	Finds the AlarmState changes for the specified Alarm.	GET <server>/services/v2/rest/alarm/id/{id}/alarmStateChanges
setAlarmState	Sets the AlarmState to the specified value.	POST <server>/services/v2/rest/alarm/setAlarmState
toString	Generates a string representing a specified Alarm.	N/A
update	Updates an existing Alarm.	POST <server>/services/v2/rest/alarm/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using Alarm Collections.

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/alarm/bulk/create
bulkDelete	POST <server>/services/v2/rest/alarm/bulk/delete
bulkSave	POST <server>/services/v2/rest/alarm/bulk/save
bulkUpdate	POST <server>/services/v2/rest/alarm/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AlarmBridge.html>

Create an Alarm

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def alarm = new Alarm(
    name: "Test create alarm",
    severity: 50,
    asset: new AssetReference(id: "Vending Machine 5000||A36") // use
asset alternate id
)
def result = alarmBridge.create(alarm)
assert result.successful

/*  the id is not populated, as the Alarm is created in an asynchronous
fashion */
assert !result.succeeded[0].id

/*  if further alarm operations are needed, the client must poll for the
alarm creation */
def numberForRetries = 47
Alarm createdAlarm = null
while (!createdAlarm || numberForRetries-- > 0) {
  Thread.sleep(1000)
  createdAlarm = alarmBridge.find(result.succeeded[0].ref)
}

assert createdAlarm
```

Find an Alarm by Alternate ID

```
import static com.axeda.sdk.v2.dsl.Bridges.*

//  alternate id is : model_number||asset_serial_number||alarm_name
def foundAlarm = alarmBridge.find("Vending Machine 5000||A36||Low stock")

assert foundAlarm
```

Find an Alarm by Criteria

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def foundAlarms = alarmBridge.find(new AlarmCriteria(name: "*", state:
AlarmState.STARTED))

assert foundAlarms.alarms.size() >= 1
```

Find an Alarm by System ID

```
import com.axeda.drm.sdk.customobject.*
import static com.axeda.sdk.v2.dsl.Bridges.*

```

```
String alarmId = Call.parameters.alarmId

def foundAlarm = alarmBridge.findById(alarmId)
assert foundAlarm
```

Find One Alarm

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def foundAlarm = alarmBridge.findOne(new AlarmCriteria(name: "*", state:
AlarmState.STARTED))

assert foundAlarm
```

Find Historical Alarms

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def foundAlarms = alarmBridge.findHistoricalAlarms(new
HistoricalAlarmCriteria(name: "Low change"))

assert foundAlarms.alarms.size() >= 1
```

Save an Alarm

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def alarm = new Alarm(
    name: "Test saved alarm",
    severity: 50,
    asset: new AssetReference(id: "Vending Machine 5000||A36")
)

def result = alarmBridge.save(alarm)
assert result.successful
```

Set Alarm State

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def stateResult = alarmBridge.setAlarmState([new AlarmReference(id:
"Vending Machine 5000||A36||Low stock")], AlarmState.ACKNOWLEDGED, "I
noticed it")
assert stateResult.successful
```

Update an Alarm

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

/*  update the severity of the initial alarm  */
```

```
def updateResult = alarmBridge.update(
    new Alarm(
        id: "Vending Machine 5000||A36||Low change", //  use the alternate
id of the alarm
        severity: 30
    ))
assert updateResult.successful
```

Bulk Create

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

/*  create the alarms */
def alarm1 = new Alarm(
    name: "Bulk alarm 1",
    severity: 50,
    asset: new AssetReference(id: "Vending Machine 5000||A36")
)

def alarm2 = new Alarm(
    name: "Bulk alarm 2",
    severity: 60,
    asset: new AssetReference(id: "Vending Machine 5000||D42")
)

def result = alarmBridge.create([alarm1, alarm2])
assert result.successful
assert result.succeeded.size() == 2
```

Bulk Save

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

/*  create the alarms */
def alarm1 = new Alarm(
    name: "Bulk alarm 1",
    severity: 60,
    asset: new AssetReference(id: "Vending Machine 5000||A36")
)

def alarm2 = new Alarm(
    name: "Bulk alarm 2",
    severity: 70,
    asset: new AssetReference(id: "Vending Machine 5000||D42")
)

def result = alarmBridge.save([alarm1, alarm2])
assert result.successful
assert result.succeeded.size() == 2
```

Bulk Update

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

/*  update the alarms using the alternate ids  */
```

```
def updateAlarm1 = new Alarm(
    id: "Vending Machine 5000||A36||Low change",
    notes: "change is needed"
)

def updateAlarm2 = new Alarm(
    id: "Vending Machine 5000||A36||Low stock",
    notes: "stock is almost 0",
    severity: 100
)

def updateResult = alarmBridge.update([updateAlarm1, updateAlarm2])
assert updateResult.successful
assert updateResult.succeeded.size() == 2

return updateResult
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AlarmBridge.html>

Count Alarms

Endpoint	/services/v2/rest/alarm/count
Verb(s)	POST
Input Object	AlarmCriteria
Output Object	

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AlarmCriteria xmlns="http://www.axeda.com/services/v2">

  <name>*1367590437960*</name>
</AlarmCriteria>
```

Create an Alarm

Endpoint	/services/v2/rest/alarm
Verb(s)	PUT
Input Object	Alarm
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Alarm xmlns="http://www.axeda.com/services/v2">
  <name>Alarm name 1366988462984</name>
  <asset systemId="19"/>
  <severity>100</severity>
  <state>STARTED</state>
  <sourceType>DEFAULT</sourceType>
  <sourceDetails>source details1366988462984</sourceDetails>

  <description>description1366988462984</description>
  <extendedData>extended data1366988462984</extendedData>
  <date>2013-04-26T11:01:02.984-04:00</date>
  <active>true</active>
  <dataItemName>data item1366988462984</dataItemName>
  <dataItemValue>data item value1366988462984</dataItemValue>

  <mobileLocation systemId="2"/>
  <notes>notes 1366988462984</notes>
</Alarm>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>model no 13669796700250
||assetSerialNumber1366979695798b||Alarm name
```

```
1366988462984</v2:ref>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Update an Alarm

Endpoint	/services/v2/rest/alarm
Verb(s)	PUT
Input Object	Alarm
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Alarm xmlns="http://www.axeda.com/services/v2">
  <name>Alarm name 1366988472120</name>
  <asset systemId="18"/>
  <severity>100</severity>
  <state>STARTED</state>
  <sourceType>DEFAULT</sourceType>
  <sourceDetails>source details1366988472120</sourceDetails>

  <description>description1366988472120</description>
  <extendedData>extended data1366988472120</extendedData>
  <date>2013-04-26T11:01:12.120-04:00</date>
  <active>true</active>
  <dataItemName>data item1366988472120</dataItemName>
  <dataItemValue>data item value1366988472120</dataItemValue>

  <mobileLocation systemId="2"/>
  <notes>notes 1366988472120</notes>
</Alarm>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>model no 13669796700250
||assetSerialNumber1366979695798a||Alarm name
1366988472120</v2:ref>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find an Alarm by Criteria

Endpoint	/services/v2/rest/alarm/find
Verb(s)	POST

Input Object	AlarmCriteria
Output Object	FindAlarmResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AlarmCriteria xmlns="http://www.axeda.com/services/v2"
pageSize="10" pageNumber="1">
  <assetId>408</assetId>
  <severity xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="BetweenNumericQuery">
    <value1>0</value1>
    <value2>200</value2>
  </severity>
  <name>Alarm name 1367590468167a</name>
  <dataItemName/>
  <dataItemValue/>
</AlarmCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindAlarmResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="1">

  <v2:criteria xsi:type="v2:AlarmCriteria" pageSize="10"
pageNumber="1">
    <v2:assetId>408</v2:assetId>
    <v2:severity xsi:type="v2:BetweenNumericQuery">
      <v2:value1>0</v2:value1>
      <v2:value2>200</v2:value2>
    </v2:severity>
    <v2:name>Alarm name 1367590468167a</v2:name>
    <v2:dataItemName/>
    <v2:dataItemValue/>
    <v2:states/>
  </v2:criteria>
  <v2:alarms>
    <v2:alarm xsi:type="v2:Alarm"
id="model_nr_NCAII13675903807730||13675903808260||Alarm name
1367590468167a" systemId="28" label="Alarm name 1367590468167a"
detail="DEFAULT" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/alarm/28">

      <v2:name>Alarm name 1367590468167a</v2:name>
      <v2:asset id="model_nr_NCAII13675903807730||
13675903808260" systemId="408"
label="model_nr_NCAII13675903807730" detail="13675903808260"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/408"/>
      <v2:severity>100</v2:severity>
      <v2:state>STARTED</v2:state>
      <v2:sourceType>DEFAULT</v2:sourceType>
      <v2:sourceDetails>source
details1367590468167a</v2:sourceDetails>
      <v2:description>
description1367590468167a</v2:description>
      <v2:extendedData>extended
data1367590468167a</v2:extendedData>
      <v2:date>2013-05-03T10:14:28.167-04:00</v2:date>
      <v2:active>true</v2:active>
```

```
      <v2:dataItemName>data
item1367590468167a</v2:dataItemName>
      <v2:dataItemValue>data item
value1367590468167a</v2:dataItemValue>
      <v2:mobileLocation id="lat:38.0||long:38.0" systemId="4"
label="38.0,38.0" detail="408"/>
      <v2:notes>notes 1367590468167a</v2:notes>
      <v2:modelAlarm systemId="28" label="Alarm name
1367590468167a" detail="description1367590468167a"/>
      <v2:notesUpdatedBy>admin</v2:notesUpdatedBy>
      <v2:notesUpdateDate>2013-05-03T10:14:28.167-04:00</v2:not
esUpdateDate>
      <v2:stateUpdateDate>2013-05-03T10:14:28.167-04:00</v2:sta
teUpdateDate>
    </v2:alarm>
  </v2:alarms>
</v2:FindAlarmResult>
```

Find Historical Alarms	
Endpoint	/services/v2/rest/alarm/findInHistory
Verb(s)	POST
Input Object	HistoricalAlarmCriteria
Output Object	FindAlarmResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<HistoricalAlarmCriteria
xmlns="http://www.axeda.com/services/v2" pageSize="10"
pageNumber="1">
  <assetId>408</assetId>
  <name>*1367590487600*</name>
</HistoricalAlarmCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindAlarmResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="2">

  <v2:criteria xsi:type="v2:HistoricalAlarmCriteria"
pageSize="10" pageNumber="1">
    <v2:assetId>408</v2:assetId>
    <v2:name>*1367590487600*</v2:name>
    <v2:states/>
  </v2:criteria>
  <v2:alarms>
    <v2:alarm xsi:type="v2:Alarm"
id="model_nr_NCAII13675903807730||13675903808260||Alarm name
1367590487600a" systemId="30" label="Alarm name 1367590487600a"
detail="DEFAULT" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/alarm/30">

      <v2:name>Alarm name 1367590487600a</v2:name>
      <v2:asset id="model_nr_NCAII13675903807730||
13675903808260" systemId="408"
label="model_nr_NCAII13675903807730" detail="13675903808260"
```

```
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/408"/>
      <v2:severity>100</v2:severity>
      <v2:state>STARTED</v2:state>
      <v2:sourceType>DEFAULT</v2:sourceType>
      <v2:sourceDetails>source
details1367590487600a</v2:sourceDetails>
      <v2:description>description1367590487600a</v2:description>

      <v2:extendedData>extended
data1367590487600a</v2:extendedData>
      <v2:date>2013-05-03T10:14:47.600-04:00</v2:date>
      <v2:active>true</v2:active>
      <v2:dataItemName>data item1367590487600a</v2:dataItemName>

      <v2:dataItemValue>data item
value1367590487600a</v2:dataItemValue>
      <v2:mobileLocation id="lat:38.0||long:38.0" systemId="4"
label="38.0,38.0" detail="408"/>
      <v2:notes>notes 1367590487600a</v2:notes>
      <v2:modelAlarm systemId="30" label="Alarm name
1367590487600a" detail="description1367590487600a"/>
      <v2:notesUpdatedBy>admin</v2:notesUpdatedBy>
      <v2:notesUpdateDate>2013-05-03T10:14:47.600-04:00</v2:note
sUpdateDate>
      <v2:stateUpdateDate>2013-05-03T10:14:47.600-04:00</v2:stat
eUpdateDate>
    </v2:alarm>
    <v2:alarm xsi:type="v2:Alarm"
id="model_nr_NCAII13675903807730||13675903808260||Alarm name
1367590487600b" systemId="31" label="Alarm name 1367590487600b"
detail="DEFAULT" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/alarm/31">

      <v2:name>Alarm name 1367590487600b</v2:name>
      <v2:asset id="model_nr_NCAII13675903807730||
13675903808260" systemId="408"
label="model_nr_NCAII13675903807730" detail="13675903808260"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/408"/>
      <v2:severity>100</v2:severity>
      <v2:state>STARTED</v2:state>
      <v2:sourceType>DEFAULT</v2:sourceType>
      <v2:sourceDetails>source
details1367590487600b</v2:sourceDetails>
      <v2:description>description1367590487600b</v2:description>

      <v2:extendedData>extended
data1367590487600b</v2:extendedData>
      <v2:date>2013-05-03T10:14:56.582-04:00</v2:date>
      <v2:active>true</v2:active>
      <v2:dataItemName>data item1367590487600b</v2:dataItemName>

      <v2:dataItemValue>data item
value1367590487600b</v2:dataItemValue>
      <v2:mobileLocation id="lat:38.0||long:38.0" systemId="4"
label="38.0,38.0" detail="408"/>
      <v2:notes>notes 1367590487600b</v2:notes>
      <v2:modelAlarm systemId="31" label="Alarm name
1367590487600b" detail="description1367590487600b"/>
      <v2:notesUpdatedBy>admin</v2:notesUpdatedBy>
```

```
      <v2:notesUpdateDate>2013-05-03T10:14:56.582-04:00</v2:note
  sUpdateDate>
      <v2:stateUpdateDate>2013-05-03T10:14:56.582-04:00</v2:stat
  eUpdateDate>
    </v2:alarm>
  </v2:alarms>
</v2:FindAlarmResult>
```

Find One Alarm

Endpoint	/services/v2/rest/alarm/findOne
Verb(s)	POST
Input Object	AlarmCriteria
Output Object	Alarm

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AlarmCriteria xmlns="http://www.axeda.com/services/v2"
pageSize="10" pageNumber="1">
  <assetId>408</assetId>
</AlarmCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:Alarm xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="model_nr_NCAII13675903807730||13675903808260||Alarm name
1367590416720" systemId="21" label="Alarm name 1367590416720"
detail="DEFAULT" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/alarm/21">
  <v2:name>Alarm name 1367590416720</v2:name>
  <v2:asset id="model_nr_NCAII13675903807730||13675903808260"
systemId="408" label="model_nr_NCAII13675903807730"
detail="13675903808260" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/408"/>

  <v2:severity>100</v2:severity>
  <v2:state>STARTED</v2:state>
  <v2:sourceType>DEFAULT</v2:sourceType>
  <v2:sourceDetails>source details1367590416720</v2:sourceDetails>

  <v2:description>description1367590416720</v2:description>
  <v2:extendedData>extended data1367590416720</v2:extendedData>

  <v2:date>2013-05-03T10:13:36.720-04:00</v2:date>
  <v2:active>true</v2:active>
  <v2:dataItemName>data item1367590416720</v2:dataItemName>
  <v2:dataItemValue>data item
value1367590416720</v2:dataItemValue>
  <v2:mobileLocation id="lat:38.0||long:38.0" systemId="4"
label="38.0,38.0" detail="408"/>
  <v2:notes>notes 1367590416720</v2:notes>
  <v2:modelAlarm systemId="21" label="Alarm name 1367590416720"
detail="description1367590416720"/>
  <v2:notesUpdatedBy>admin</v2:notesUpdatedBy>
  <v2:notesUpdateDate>2013-05-03T10:13:36.720-04:00</v2:notesUpdat
eDate>
  <v2:stateUpdateDate>2013-05-03T10:13:36.720-04:00</v2:stateUpdat
```

```
eDate>
</v2:Alarm>
```

Set an Alarm State

Endpoint	/services/v2/rest/alarm/setAlarmState
Verb(s)	POST
Input Object	AlarmCollection
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AlarmCollection xmlns="http://www.axeda.com/services/v2">
  <alarm xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Alarm" id="model_nr_NCAII13675903807730||13675903808260
||Alarm name 1367590507009" systemId="32" label="Alarm name
1367590507009" detail="DEFAULT" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/alarm/32">

    <name>Alarm name 1367590507009</name>
    <asset id="model_nr_NCAII13675903807730||13675903808260"
systemId="408" label="model_nr_NCAII13675903807730"
detail="13675903808260" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/408"/>

    <severity>100</severity>
    <state>STARTED</state>
    <sourceType>DEFAULT</sourceType>
    <sourceDetails>source details1367590507009</sourceDetails>

    <description>description1367590507009</description>
    <extendedData>extended data1367590507009</extendedData>
    <date>2013-05-03T10:15:07.009-04:00</date>
    <active>true</active>
    <dataItemName>data item1367590507009</dataItemName>
    <dataItemValue>data item value1367590507009</dataItemValue>

    <mobileLocation id="lat:38.0||long:38.0" systemId="4"
label="38.0,38.0" detail="408"/>
    <notes>notes 1367590507009</notes>
    <modelAlarm systemId="32" label="Alarm name 1367590507009"
detail="description1367590507009"/>
    <notesUpdatedBy>admin</notesUpdatedBy>
    <notesUpdateDate>2013-05-03T10:15:07.009-04:00</notesUpdateDa
te>
    <stateUpdateDate>2013-05-03T10:15:07.009-04:00</stateUpdateDa
te>
  </alarm>
</AlarmCollection>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
```

```
    <v2:success>
      <v2:ref>32</v2:ref>
      <v2:id>32</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AlarmBridge.html>

AssetBridge

An Asset is generally any piece of equipment that a company owns that could benefit by being remotely monitored and managed. Remote monitoring and management can reduce downtime, decrease service calls, and improve customer satisfaction. With Axeda, Assets can be any Property, Plant, and Equipment (PPE) devices you would like to monitor with the Axeda Platform.

In order for the Axeda Platform to monitor an Asset, communication between the Axeda Platform and the Asset must exist. This communication does not need to be constant; the timing of the communication can be set up to suit the needs of your organization. Assets can communicate with the Axeda Platform over a variety of communication media, including wireless networks.

An Asset represents one instance of a Model, and requires a reference to a Model (the Model Name), as well as an individual serial number for the Asset.

Assets can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating Assets through the API using the AssetBridge object.

The AssetBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for Asset objects. You can also use the AssetBridge to register Assets, and to list recently viewed Assets.

Asset Details

The AssetDetails object contains fields that require additional look-up in the database, and that therefore may affect performance. Because it is not required to include these fields with every Asset operation, they are split off into an AssetDetails object that can be queried only when necessary. AssetDetails fields include gateway parameters and references, asset groups, primary contacts, time zone, and token.

The AssetBridge addDetail method can be used to return an Asset with its details fields populated with data.

Asset Criteria

The AssetBridge Find By Criteria method takes an AssetCriteria object as input, and returns a FindAssetResult object that lists (as a collection) the Assets with the specified criteria. The FindAssetResult object is always paginated.

Asset search criteria can include any set of valid Asset or AssetDetails object properties. For a full description of AssetCriteria properties and methods, see the Axeda API Specification.

Methods

The following table provides a brief overview of the AssetBridge methods, along with available REST calls. For a full description of AssetBridge methods, see the Axeda API Specification.

Method	Description	REST Call
addDetail	Returns a specified Asset with its details fields	N/A

	populated with data.	
countAssetsByCriteria	Returns the number of Assets that meet the specified criteria.	POST <server>/services/v2/rest/asset/countAssetsByCriteria
create	Creates a new Asset.	PUT <server>/services/v2/rest/asset **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/asset
delete	Deletes an Asset.	DELETE <server>/services/v2/rest/asset/id/{id}
find (by criteria)	Finds Assets based on search criteria.	POST <server>/services/v2/rest/asset/find
find (by IDs)	Finds Assets based on a list of identifiers.	POST <server>/services/v2/rest/asset/findByIds
find (by alternate ID)	Finds an Asset based on the alternate identifier.	GET <server>/services/v2/rest/asset/id/{id}
findById	Finds an Asset based on its platform identifier.	GET <server>/services/v2/rest/asset/id/{id}
findOne	Returns the first Asset found that meets specified search criteria.	POST <server>/services/v2/rest/asset/findOne
flagRecentlyViewed	Flags the specified asset as recently viewed.	N/A
getRecentlyViewed	Returns a list of the most recently viewed assets.	GET <server>/services/v2/rest/asset/recentlyViewed
register	Registers the specified asset.	POST <server>/services/v2/rest/asset/id/{id}/register
toString	Generates a string representing a specified Asset.	N/A
update	Updates an existing Asset.	POST <server>/services/v2/rest/asset/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using Asset Collections.

Operation	REST Call

bulkCreate	POST <server>/services/v2/rest/asset/bulk/create
bulkDelete	POST <server>/services/v2/rest/asset/bulk/delete
bulkSave	POST <server>/services/v2/rest/asset/bulk/save
bulkUpdate	POST <server>/services/v2/rest/asset/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetBridge.html>

Create an Asset - Sample 1

```
import com.axeda.services.v2.*

//ensure a model exists
Model model = bridges.modelBridge.find("FooModel")
if(!model)
{
  //model doesn't exist
  model = new Model(modelNumber:"FooModel")
  ExecutionResult modelCreateResult = bridges.modelBridge.create(model)

  if(!modelCreateResult.succeeded)
  {
    //something unexpected happened! Return the results to the caller

    return modelCreateResult
  }
}

Asset asset = new
Asset(name:"BarAsset",serialNumber:"BarAsset",model:model)
ExecutionResult createAssetResult = bridges.assetBridge.create(asset)


if(!createAssetResult.succeeded)
{
  //something unexpected happened! Return the results to the caller
  //let's see if this asset just already exists maybe?
  if(createAssetResult?.failures?.any { it?.code == "NOT_UNIQUE" &&
it.sourceOfFailure == "asset.serialNumber" })
  {
    //aha - it must have already existed!
    return bridges.assetBridge.find("FooModel||BarAsset")
  }
  return createAssetResult
}

return bridges.assetBridge.findById(createAssetResult.successful[0].id)
```

Create an Asset - Sample 2

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.Asset
import com.axeda.services.v2.AssetCriteria
import com.axeda.services.v2.Model
import com.axeda.services.v2.ModelCriteria

/***************
 *
 * Here we use a previously created model to create an Asset.
 *
```

```
 * The individual things we are concerned with are now called Assets, not
Devices.
 *
 * You might notice that asset.model has a property of detail, not
name.  This is because
 * asset.model returns a ModelReference, not a Model.  In many cases all
that is needed from
 * the model of an Asset is the name or some other trivial data, as
opposed to situations where
 * the actual object is needed.  For this reason, a shallow copy of the
model is provided as a
 * ModelReference, which contains the systemId and detail as the name.

 *
 * You can find out more about any object by calling its dump() method,
which prints out all its properties.
 *
 * **************/

def response = ""

ModelCriteria crit = new ModelCriteria()
crit.modelNumber = "Fo*"
def modelresults = modelBridge.find(crit)

Model model

if (modelresults.totalCount == 0){
    throw new Exception("No Model with that Name")
} else {
    model = modelresults.models[0]
}

// model = modelBridge.find("FooModel")  // this would have worked also


def asset = new
Asset([name:"BarAsset",serialNumber:"BarAsset",model:model])

def result = assetBridge.create(asset)

response = "name is " + asset.serialNumber + ", model is " + asset.model


// prints out all the properties of ModelReference
logger.info(asset.model.dump())

return ['Content-Type': 'application/text', 'Content': response]
```

Find an Asset by Criteria

```
import com.axeda.services.v2.*

AssetCriteria assetCriteria = new AssetCriteria()
assetCriteria.serialNumber = "BarAsset"
return bridges.assetBridge.find(assetCriteria)

```

Upate an Asset

```
import com.axeda.services.v2.*

   AssetCriteria assetCriteria = new AssetCriteria()
   assetCriteria.serialNumber = "BarAsset"
   Asset asset = bridges.assetBridge.findOne(assetCriteria)
   asset.setDescription("i changed the description")
   ExecutionResult result = bridges.assetBridge.update(asset)
   if(!result.successful)
   {
     //uh oh!
     return result
   }
   return bridges.assetBridge.findOne(assetCriteria) // find it back again
using asset id to demonstrate the updated description
```

Delete an Asset

```
import com.axeda.services.v2.*

return
bridges.assetBridge.delete(bridges.assetBridge.find("FooModel||BarAsset"))
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetBridge.html>

Create an Asset

Endpoint	/services/v2/rest/asset
Verb(s)	PUT
Input Object	Asset
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Asset xmlns="http://www.axeda.com/services/v2">
  <name>assetName1367256270449</name>
  <serialNumber>assetSerialNumber1367256270449</serialNumber>

  <buildVersion>1367256270449</buildVersion>
  <timeZone>America/New_York</timeZone>
  <description>assetDescription1367256270449</description>
  <sharedKey>0102030405</sharedKey>
  <pingRate>1798</pingRate>
  <tokenInvalid>true</tokenInvalid>
  <language>RO</language>
  <agentVersion>6.6</agentVersion>
  <alternateId>altId1367256270449</alternateId>
  <address>test address</address>
  <token>test token</token>
  <backupToken>backToken</backupToken>
  <categoryCode>1</categoryCode>
  <retryCount>15</retryCount>
  <retryInterval>3</retryInterval>
  <secondarySerialNumber>
secSN1367256270449</secondarySerialNumber>
  <protocolId>1</protocolId>
  <model systemId="780"/>
  <customer systemId="151"/>
  <location systemId="300"/>
  <gateways>
    <gateway systemId="1417"/>
    <gateway systemId="1418"/>
  </gateways>
</Asset>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>model no 13672562588220
||assetSerialNumber1367256270449</v2:ref>
      <v2:id>1426</v2:id>
    </v2:success>
```

```
    </v2:succeeded>
    <v2:failures/>
  </v2:ExecutionResult>
```

Update an Asset

Endpoint	/services/v2/rest/asset/id/1427
Verb(s)	POST
Input Object	Asset
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Asset xmlns="http://www.axeda.com/services/v2" systemId="1427">

  <name>assetName1367256272826a</name>
  <serialNumber>assetSerialNumber1367256272011</serialNumber>

  <buildVersion>1367256272826a</buildVersion>
  <timeZone>America/New_York</timeZone>
  <description>assetDescription1367256272826a</description>

  <sharedKey>0102030405</sharedKey>
  <pingRate>530</pingRate>
  <tokenInvalid>true</tokenInvalid>
  <language>RO</language>
  <agentVersion>6.6</agentVersion>
  <alternateId>altId1367256272826a</alternateId>
  <address>test address</address>
  <token>test token</token>
  <backupToken>backToken</backupToken>
  <categoryCode>1</categoryCode>
  <retryCount>15</retryCount>
  <retryInterval>3</retryInterval>
  <secondarySerialNumber>
secSN1367256272826a</secondarySerialNumber>
  <protocolId>1</protocolId>
  <model systemId="780"/>
  <customer systemId="152"/>
  <location systemId="302"/>
  <gateways>
    <gateway systemId="1419"/>
    <gateway systemId="1420"/>
    <gateway systemId="1421"/>
  </gateways>
</Asset>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>1427</v2:ref>
      <v2:id>1427</v2:id>
    </v2:success>
```

```
    </v2:succeeded>
    <v2:failures/>
  </v2:ExecutionResult>
```

Delete an Asset

Endpoint	/services/v2/rest/asset/id/286
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>286</v2:ref>
      <v2:id>286</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find One Asset

Endpoint	/services/v2/rest/asset/findOne
Verb(s)	POST
Input Object	AssetCriteria
Output Object	Asset

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AssetCriteria xmlns="http://www.axeda.com/services/v2">
  <serialNumber>assetSerialNumber1367590917847</serialNumber>

</AssetCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:Asset xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="model no
13675909002550||assetSerialNumber1367590917847" systemId="424"
label="model no 13675909002550"
detail="assetSerialNumber1367590917847" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/424">


  <v2:name>assetName1367590917847</v2:name>
  <v2:serialNumber>
assetSerialNumber1367590917847</v2:serialNumber>
  <v2:buildVersion>1367590917847</v2:buildVersion>
  <v2:timeZone>America/New_York</v2:timeZone>
```

```
  <v2:description>assetDescription1367590917847</v2:description>

  <v2:sharedKey>0102030405</v2:sharedKey>
  <v2:pingRate>469</v2:pingRate>
  <v2:missing>false</v2:missing>
  <v2:offline>false</v2:offline>
  <v2:priority>1</v2:priority>
  <v2:EMessageAsset>true</v2:EMessageAsset>
  <v2:tokenInvalid>false</v2:tokenInvalid>
  <v2:language>test</v2:language>
  <v2:agentVersion>6.6</v2:agentVersion>
  <v2:alternateId>altId1367590917847</v2:alternateId>
  <v2:address>test address</v2:address>
  <v2:categoryCode>1</v2:categoryCode>
  <v2:retryCount>15</v2:retryCount>
  <v2:retryInterval>3</v2:retryInterval>
  <v2:secondarySerialNumber>
secSN1367590917847</v2:secondarySerialNumber>
  <v2:protocolId>1</v2:protocolId>
  <v2:model id="model no 13675909002550" systemId="77"
label="standalone" detail="model no 13675909002550"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/77"/>
  <v2:customer id="orgName13675909002550" systemId="77"
label="orgName13675909002550"
detail="orgDescription13675909002550" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/77"/>

  <v2:location id="orgName13675909002550||loc name 13675909002550"
systemId="208" label="loc name 13675909002550"
detail="orgName13675909002550" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/208"/>

  <v2:condition xsi:type="v2:Condition" systemId="3" label="Good"
detail="1">
    <v2:name>Good</v2:name>
  </v2:condition>
  <v2:properties/>
  <v2:gateways>
    <v2:gateway systemId="410"
label="assetDescription13675909002550"
detail="assetDescription13675909002550"/>
    <v2:gateway systemId="411"
label="assetDescription13675909002551"
detail="assetDescription13675909002551"/>
  </v2:gateways>
  <v2:muted>false</v2:muted>
</v2:Asset>
```

Find an Asset by Criteria

Endpoint	/services/v2/rest/asset/find
Verb(s)	POST
Input Object	AssetCriteria
Output Object	FindAssetResult

XML Request

```xml
<?xml version="1.0" encoding="UTF-8"?>
<AssetCriteria xmlns="http://www.axeda.com/services/v2">
  <modelNumber>model no 13675909002550</modelNumber>
  <serialNumber>*1367590926993a*</serialNumber>
</AssetCriteria>
```

XML Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindAssetResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="2">

  <v2:criteria>
    <v2:modelNumber>model no 13675909002550</v2:modelNumber>

    <v2:serialNumber>*1367590926993a*</v2:serialNumber>
    <v2:propertyNames/>
  </v2:criteria>
  <v2:assets>
    <v2:asset xsi:type="v2:Asset" id="model no 13675909002550
||assetSerialNumber1367590926993a1" systemId="434" label="model
no 13675909002550" detail="assetSerialNumber1367590926993a1"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/434">

      <v2:name>assetName1367590926993a1</v2:name>
      <v2:serialNumber>
assetSerialNumber1367590926993a1</v2:serialNumber>
      <v2:buildVersion>1367590926993a1</v2:buildVersion>
      <v2:timeZone>America/New_York</v2:timeZone>
      <v2:description>
assetDescription1367590926993a1</v2:description>
      <v2:sharedKey>0102030405</v2:sharedKey>
      <v2:pingRate>39</v2:pingRate>
      <v2:missing>false</v2:missing>
      <v2:offline>false</v2:offline>
      <v2:priority>1</v2:priority>
      <v2:EMessageAsset>true</v2:EMessageAsset>
      <v2:tokenInvalid>false</v2:tokenInvalid>
      <v2:language>test</v2:language>
      <v2:agentVersion>6.6</v2:agentVersion>
      <v2:alternateId>altId1367590926993a1</v2:alternateId>

      <v2:address>test address</v2:address>
      <v2:categoryCode>1</v2:categoryCode>
      <v2:retryCount>15</v2:retryCount>
      <v2:retryInterval>3</v2:retryInterval>
      <v2:secondarySerialNumber>
secSN1367590926993a1</v2:secondarySerialNumber>
      <v2:protocolId>1</v2:protocolId>
      <v2:model id="model no 13675909002550" systemId="77"
label="standalone" detail="model no 13675909002550"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/77"/>
      <v2:customer id="orgName13675909002550" systemId="77"
label="orgName13675909002550"
detail="orgDescription13675909002550" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/77"/>

      <v2:location id="orgName13675909002550||loc name
13675909002550" systemId="208" label="loc name 13675909002550"
detail="orgName13675909002550" restUrl="http://jtelarico-
```

```
dt7.axeda.com:8080/services/v2/rest/location/208"/>

      <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
        <v2:name>Good</v2:name>
      </v2:condition>
      <v2:properties/>
      <v2:gateways>
        <v2:gateway systemId="410"
label="assetDescription13675909002550"
detail="assetDescription13675909002550"/>
        <v2:gateway systemId="411"
label="assetDescription13675909002551"
detail="assetDescription13675909002551"/>
      </v2:gateways>
      <v2:muted>false</v2:muted>
    </v2:asset>
    <v2:asset xsi:type="v2:Asset" id="model no 13675909002550
||assetSerialNumber1367590926993a2" systemId="435" label="model
no 13675909002550" detail="assetSerialNumber1367590926993a2"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/435">

      <v2:name>assetName1367590926993a2</v2:name>
      <v2:serialNumber>
assetSerialNumber1367590926993a2</v2:serialNumber>
      <v2:buildVersion>1367590926993a2</v2:buildVersion>
      <v2:timeZone>America/New_York</v2:timeZone>
      <v2:description>
assetDescription1367590926993a2</v2:description>
      <v2:sharedKey>0102030405</v2:sharedKey>
      <v2:pingRate>40</v2:pingRate>
      <v2:missing>false</v2:missing>
      <v2:offline>false</v2:offline>
      <v2:priority>1</v2:priority>
      <v2:EMessageAsset>true</v2:EMessageAsset>
      <v2:tokenInvalid>false</v2:tokenInvalid>
      <v2:language>test</v2:language>
      <v2:agentVersion>6.6</v2:agentVersion>
      <v2:alternateId>altId1367590926993a2</v2:alternateId>

      <v2:address>test address</v2:address>
      <v2:categoryCode>1</v2:categoryCode>
      <v2:retryCount>15</v2:retryCount>
      <v2:retryInterval>3</v2:retryInterval>
      <v2:secondarySerialNumber>
secSN1367590926993a2</v2:secondarySerialNumber>
      <v2:protocolId>1</v2:protocolId>
      <v2:model id="model no 13675909002550" systemId="77"
label="standalone" detail="model no 13675909002550"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/77"/>
      <v2:customer id="orgName13675909002550" systemId="77"
label="orgName13675909002550"
detail="orgDescription13675909002550" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/77"/>

      <v2:location id="orgName13675909002550||loc name
13675909002550" systemId="208" label="loc name 13675909002550"
detail="orgName13675909002550" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/208"/>
```

```
      <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
        <v2:name>Good</v2:name>
      </v2:condition>
      <v2:properties/>
      <v2:gateways>
        <v2:gateway systemId="410"
label="assetDescription13675909002550"
detail="assetDescription13675909002550"/>
        <v2:gateway systemId="411"
label="assetDescription13675909002551"
detail="assetDescription13675909002551"/>
      </v2:gateways>
      <v2:muted>false</v2:muted>
    </v2:asset>
  </v2:assets>
</v2:FindAssetResult>
```

Get Recently Viewed Assets

Endpoint	/services/v2/rest/asset/recentlyViewed
Verb(s)	GET
Input Object	
Output Object	AssetCollection

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:AssetCollection xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <v2:asset xsi:type="v2:Asset" id="model no 13675909002550
||assetSerialNumber1367590908260" systemId="416" label="model
no 13675909002550" detail="assetSerialNumber1367590908260"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/416">

    <v2:name>assetName1367590908260</v2:name>
    <v2:serialNumber>
assetSerialNumber1367590908260</v2:serialNumber>
    <v2:buildVersion>1367590908260</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367590908260</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>1028</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367590908260</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
```

```
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367590908260</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675909002550" systemId="77"
label="standalone" detail="model no 13675909002550"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/77"/>
    <v2:customer id="orgName13675909002550" systemId="77"
label="orgName13675909002550"
detail="orgDescription13675909002550"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/77"/>
    <v2:location id="orgName13675909002550||loc name
13675909002550" systemId="208" label="loc name 13675909002550"
detail="orgName13675909002550" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/208"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="410"
label="assetDescription13675909002550"
detail="assetDescription13675909002550"/>
      <v2:gateway systemId="411"
label="assetDescription13675909002551"
detail="assetDescription13675909002551"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675902795970
||assetSerialNumber1367590303800" systemId="399" label="model
no 13675902795970" detail="assetSerialNumber1367590303800"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/399">

    <v2:name>assetName1367590303800</v2:name>
    <v2:serialNumber>
assetSerialNumber1367590303800</v2:serialNumber>
    <v2:buildVersion>1367590303800</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367590303800</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>747</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367590303800</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
```

```
secSN1367590303800</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675902795970" systemId="73"
label="standalone" detail="model no 13675902795970"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/73"/>
    <v2:customer id="orgName13675902795970" systemId="75"
label="orgName13675902795970"
detail="orgDescription13675902795970"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/75"/>
    <v2:location id="orgName13675902795970||loc name
13675902795970" systemId="204" label="loc name 13675902795970"
detail="orgName13675902795970" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/204"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="372"
label="assetDescription13675902795970"
detail="assetDescription13675902795970"/>
      <v2:gateway systemId="373"
label="assetDescription13675902795971"
detail="assetDescription13675902795971"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675902795970
||assetSerialNumber1367590302824" systemId="398" label="model
no 13675902795970" detail="assetSerialNumber1367590302824"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/398">

    <v2:name>assetName1367590302824</v2:name>
    <v2:serialNumber>
assetSerialNumber1367590302824</v2:serialNumber>
    <v2:buildVersion>1367590302824</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367590302824</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>1622</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367590302824</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367590302824</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
```

```
    <v2:model id="model no 13675902795970" systemId="73"
label="standalone" detail="model no 13675902795970"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/73"/>
    <v2:customer id="orgName13675902795970" systemId="75"
label="orgName13675902795970"
detail="orgDescription13675902795970"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/75"/>
    <v2:location id="orgName13675902795970||loc name
13675902795970" systemId="204" label="loc name 13675902795970"
detail="orgName13675902795970" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/204"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="372"
label="assetDescription13675902795970"
detail="assetDescription13675902795970"/>
      <v2:gateway systemId="373"
label="assetDescription13675902795971"
detail="assetDescription13675902795971"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675902795970
||assetSerialNumber1367590301207" systemId="397" label="model
no 13675902795970" detail="assetSerialNumber1367590301207"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/397">

    <v2:name>assetName1367590301207</v2:name>
    <v2:serialNumber>
assetSerialNumber1367590301207</v2:serialNumber>
    <v2:buildVersion>1367590301207</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367590301207</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>206</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367590301207</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367590301207</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675902795970" systemId="73"
label="standalone" detail="model no 13675902795970"
```

```
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/73"/>
    <v2:customer id="orgName13675902795970" systemId="75"
label="orgName13675902795970"
detail="orgDescription13675902795970"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/75"/>
    <v2:location id="orgName13675902795970||loc name
13675902795970" systemId="204" label="loc name 13675902795970"
detail="orgName13675902795970" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/204"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="372"
label="assetDescription13675902795970"
detail="assetDescription13675902795970"/>
      <v2:gateway systemId="373"
label="assetDescription13675902795971"
detail="assetDescription13675902795971"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675898801750
||assetSerialNumber1367590031315" systemId="352" label="model
no 13675898801750" detail="assetSerialNumber1367590031315"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/352">

    <v2:name>assetName1367590031315</v2:name>
    <v2:serialNumber>
assetSerialNumber1367590031315</v2:serialNumber>
    <v2:buildVersion>1367590031315</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367590031315</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>1106</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367590031315</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367590031315</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675898801750" systemId="70"
label="standalone" detail="model no 13675898801750"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/70"/>
```

```
    <v2:customer id="orgName13675898801750" systemId="73"
label="orgName13675898801750"
detail="orgDescription13675898801750"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/73"/>
    <v2:location id="orgName13675898801750||loc name
13675898801750" systemId="200" label="loc name 13675898801750"
detail="orgName13675898801750" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/200"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="337"
label="assetDescription13675898801751"
detail="assetDescription13675898801751"/>
      <v2:gateway systemId="336"
label="assetDescription13675898801750"
detail="assetDescription13675898801750"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675898801750
||assetSerialNumber1367590029472" systemId="351" label="model
no 13675898801750" detail="assetSerialNumber1367590029472"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/351">

    <v2:name>assetName1367590029472</v2:name>
    <v2:serialNumber>
assetSerialNumber1367590029472</v2:serialNumber>
    <v2:buildVersion>1367590029472</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367590029472</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>1057</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367590029472</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367590029472</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675898801750" systemId="70"
label="standalone" detail="model no 13675898801750"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/70"/>
    <v2:customer id="orgName13675898801750" systemId="73"
label="orgName13675898801750"
```

```
detail="orgDescription13675898801750"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/73"/>
    <v2:location id="orgName13675898801750||loc name
13675898801750" systemId="200" label="loc name 13675898801750"
detail="orgName13675898801750" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/200"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="337"
label="assetDescription13675898801751"
detail="assetDescription13675898801751"/>
      <v2:gateway systemId="336"
    label="assetDescription13675898801750"
    detail="assetDescription13675898801750"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675898801750
    ||assetSerialNumber1367590027589" systemId="350" label="model
    no 13675898801750" detail="assetSerialNumber1367590027589"
    restUrl="http://jtelarico-
    dt7.axeda.com:8080/services/v2/rest/asset/350">

    <v2:name>assetName1367590027589</v2:name>
    <v2:serialNumber>
    assetSerialNumber1367590027589</v2:serialNumber>
    <v2:buildVersion>1367590027589</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
    assetDescription1367590027589</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>74</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367590027589</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
    secSN1367590027589</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675898801750" systemId="70"
    label="standalone" detail="model no 13675898801750"
    restUrl="http://jtelarico-
    dt7.axeda.com:8080/services/v2/rest/model/70"/>
    <v2:customer id="orgName13675898801750" systemId="73"
    label="orgName13675898801750"
    detail="orgDescription13675898801750"
    restUrl="http://jtelarico-
```

```
dt7.axeda.com:8080/services/v2/rest/organization/73"/>
    <v2:location id="orgName13675898801750||loc name
13675898801750" systemId="200" label="loc name 13675898801750"
detail="orgName13675898801750" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/200"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="337"
label="assetDescription13675898801751"
detail="assetDescription13675898801751"/>
      <v2:gateway systemId="336"
label="assetDescription13675898801750"
detail="assetDescription13675898801750"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675897526020
||assetSerialNumber1367589761375" systemId="335" label="model
no 13675897526020" detail="assetSerialNumber1367589761375"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/335">

    <v2:name>assetName1367589761375</v2:name>
    <v2:serialNumber>
assetSerialNumber1367589761375</v2:serialNumber>
    <v2:buildVersion>1367589761375</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367589761375</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>1686</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367589761375</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367589761375</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675897526020" systemId="67"
label="standalone" detail="model no 13675897526020"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/67"/>
    <v2:customer id="orgName13675897526020" systemId="71"
label="orgName13675897526020"
detail="orgDescription13675897526020"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/71"/>
    <v2:location id="orgName13675897526020||loc name
```

```
13675897526020" systemId="196" label="loc name 13675897526020"
detail="orgName13675897526020" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/196"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="329"
label="assetDescription13675897526020"
detail="assetDescription13675897526020"/>
      <v2:gateway systemId="330"
label="assetDescription13675897526021"
detail="assetDescription13675897526021"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675890466710
||assetSerialNumber1367589071426" systemId="311" label="model
no 13675890466710" detail="assetSerialNumber1367589071426"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/311">

    <v2:name>assetName1367589071426</v2:name>
    <v2:serialNumber>
assetSerialNumber1367589071426</v2:serialNumber>
    <v2:buildVersion>1367589071426</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367589071426</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>15</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367589071426</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367589071426</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675890466710" systemId="58"
label="standalone" detail="model no 13675890466710"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/58"/>
    <v2:customer id="orgName13675890466710" systemId="65"
label="orgName13675890466710"
detail="orgDescription13675890466710"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/65"/>
    <v2:location id="orgName13675890466710||loc name
13675890466710" systemId="184" label="loc name 13675890466710"
detail="orgName13675890466710" restUrl="http://jtelarico-
```

```
dt7.axeda.com:8080/services/v2/rest/location/184"/>

    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="285"
label="assetDescription13675890466711"
detail="assetDescription13675890466711"/>
      <v2:gateway systemId="284"
label="assetDescription13675890466710"
detail="assetDescription13675890466710"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
  <v2:asset xsi:type="v2:Asset" id="model no 13675890466710
||assetSerialNumber1367589069763" systemId="310" label="model
no 13675890466710" detail="assetSerialNumber1367589069763"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/310">

    <v2:name>assetName1367589069763</v2:name>
    <v2:serialNumber>
assetSerialNumber1367589069763</v2:serialNumber>
    <v2:buildVersion>1367589069763</v2:buildVersion>
    <v2:timeZone>America/New_York</v2:timeZone>
    <v2:description>
assetDescription1367589069763</v2:description>
    <v2:sharedKey>0102030405</v2:sharedKey>
    <v2:pingRate>4</v2:pingRate>
    <v2:missing>false</v2:missing>
    <v2:offline>false</v2:offline>
    <v2:priority>1</v2:priority>
    <v2:EMessageAsset>true</v2:EMessageAsset>
    <v2:tokenInvalid>false</v2:tokenInvalid>
    <v2:language>test</v2:language>
    <v2:agentVersion>6.6</v2:agentVersion>
    <v2:alternateId>altId1367589069763</v2:alternateId>
    <v2:address>test address</v2:address>
    <v2:categoryCode>1</v2:categoryCode>
    <v2:retryCount>15</v2:retryCount>
    <v2:retryInterval>3</v2:retryInterval>
    <v2:secondarySerialNumber>
secSN1367589069763</v2:secondarySerialNumber>
    <v2:protocolId>1</v2:protocolId>
    <v2:model id="model no 13675890466710" systemId="58"
label="standalone" detail="model no 13675890466710"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/58"/>
    <v2:customer id="orgName13675890466710" systemId="65"
label="orgName13675890466710"
detail="orgDescription13675890466710"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/65"/>
    <v2:location id="orgName13675890466710||loc name
13675890466710" systemId="184" label="loc name 13675890466710"
detail="orgName13675890466710" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/184"/>
```

```
    <v2:condition xsi:type="v2:Condition" systemId="3"
label="Good" detail="1">
      <v2:name>Good</v2:name>
    </v2:condition>
    <v2:properties/>
    <v2:gateways>
      <v2:gateway systemId="285"
label="assetDescription13675890466711"
detail="assetDescription13675890466711"/>
      <v2:gateway systemId="284"
label="assetDescription13675890466710"
detail="assetDescription13675890466710"/>
    </v2:gateways>
    <v2:muted>false</v2:muted>
  </v2:asset>
</v2:AssetCollection>
```

Register an Asset

Endpoint	/services/v2/rest/asset/id/426/register
Verb(s)	POST
Input Object	
Output Object	ExecutionResult
XML Response	

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>426</v2:ref>
      <v2:id>426</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetBridge.html>

AssetConfigurationBridge

Asset Configuration enables you to create and update Asset Configuration information and apply it to one or more Assets. This section provides an overview of creating and manipulating AssetConfiguration objects through the Axeda API using the AssetConfigurationBridge object.

Asset Configuration information is stored as a series of key-value pairs, where the keys are XPath-style paths. A single key-value pair is referred to as "ConfigurationItem." The configuration of an Asset is a collection of these ConfigurationItem pairs, initially expressed as an XML document.

A typical Configuration is as follows:

```
<Config>
  <Device serial="1234"
    <Name>Foo</Name>
    <Vendor>Dell</Vendor>
  </Device>
  <Device serial="4321">
    <Name>Bar</Name>
  </Device>
 </Config>
```

This Configuration would result in the following ConfigurationItem entries:

Key	Value
/Config/Device[1]/@serial	1234
/Config/Device[1]/Name	Foo
/Config/Device[1]/Vendor	Dell
/Config/Device[2]/@serial	4321
/Config/Device[2]/Name	Bar

To convert this XML into an AssetConfiguration, use the "fromXml(String xml, String assetId)" method, which will set the Asset's configuration to the supplied values. Once this Asset Configuration has been stored, it can be modified using the appendConfigurationItems and replaceConfigurationItems methods, or by using the provided CRUD methods to create, update, or delete the AssetConfiguration itself. Additional methods enable you to validate an Asset Configuration based on the Configuration Validation rules for a Model, or based on supplied configuration XML.

Methods

The following table provides a brief overview of the available AssetConfigurationBridge methods. For a full

description of AssetConfigurationBridge methods, see the Axeda API Specification.

Method	Description
appendConfigurationItems	Appends ConfigurationItems to the supplied Asset Configuration.
create	Creates a new Asset Configuration object.
create (with validation)	Creates a new Asset Configuration object only if the supplied Configuration XML is valid based on the Configuration Validation rules for the specified Asset's Model.
create (with or without validation)	Creates a new Asset Configuration using the supplied Configuration XML for the specified Asset. A Boolean validation flag determines whether validation will take place. If validation is enabled and validation fails, the Asset Configuration will not be created.
delete	Deletes an Asset Configuration object.
deleteConfigurationItems	Deletes Asset Configuration objects based on a list of identifiers.
find (by criteria)	Finds Asset Configuration objects based on search criteria.
find (by IDs)	Finds Asset Configuration objects based on a list of identifiers.
find (by alternate ID)	Finds an Asset Configuration object based on the alternate identifier.
findAssetConfigurationByAssetId	Finds a Region object based on a supplied assetId.
findById	Finds an Asset Configuration object based on its platform identifier.
findOne	Returns the first Asset Configuration object found that meets specified search criteria.
fromXml	Converts the supplied XML into an Asset Configuration.
generateAlternateId	Returns a unique alternate identifier for the specified Asset Configuration.
getAssetConfiguration	Returns an Asset Configuration from the provided assetId, which represents a tree/sub-tree of the Asset's configuration based on the startingPath provided.
isMatchingAssetConfig	Returns True if the supplied Configuration XML matches the stored Configuration for the supplied AssetId at the supplied root path.
replaceConfigurationItems	Replaces the supplied Configuration Items in the supplied Asset Configuration.
toString	Generates a string representing a specified Asset Configuration object.
toXml	Returns a String representing the XML equivalent for the supplied Asset Configuration.
toXml (with path)	Returns a String representing the XML equivalent for the Configuration Items stored for the supplied AssetId at the supplied parent path.
update	Updates an existing Asset Configuration object.
validate	Validates the specified Asset Configuration based on the validation rules associated with the supplied assetId.
validate	Validates an Asset Configuration for the supplied Configuration XML and assetId.

(Configuration XML)	
validate (Configuration XML with parentPath)	Validates an Asset Configuration for the supplied Configuration XML and assetId at the specified parentPath.

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetConfigurationBridge.html>

Create an AssetConfiguration

```
//The following code is used to store a
Configuration:

 import com.axeda.sdk.v2.dsl.Bridges.*

   String configAsXml = """<DeviceSync>
<FolderStructure><Folder folderName="a"/><Folder
folderName="b"/></FolderStructure><LabInformation
city="someCity"/></DeviceSync>"""
   String assetId = 1

   AssetConfiguration configToCreate =
assetConfigurationBridge.fromXml(configAsXml,
assetId);
   //perform the save (create and update do the
same thing)
   ExecutionResult result =
assetConfigurationBridge.create(configToCreate)
```

Find an AssetConfiguration

```
//The following code is used to find a
Configuration and return it as XML:

 AssetConfiguration config =
assetConfigurationBridge.getAssetConfiguration(ass
etId,"");
   return assetConfigurationBridge.toXml(config);

//The following code is used to determine if the
Configuration for assetId 1 includes
//"/Config/AssetConfiguration":

 String path = "/Config/AssetConfiguration"
   assetConfigurationBridge.isMatchingAssetConfig(
configAsXml, "/", assetId)
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetConfigurationBridge.html>

AssetGroupBridge

An Asset Group is a set of Assets that can be associated with one another for business purposes. Each Asset Group can contain Assets, one other Asset Group, or a combination of Assets and one other Asset Group. Although an Asset group can be assigned to only one other Asset group (its "parent"), a single Asset can be assigned to multiple Asset Groups. An Asset Group within another Asset Group is considered a "child" of that "parent."

Asset Groups can be created and manipulated using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating Asset Groups through the API using the AssetGroupBridge object.

The AssetGroupBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for Asset objects. The getAssets method returns the Assets that belong to an Asset Group, and the getChildren method returns child Asset Groups.

Methods

The following table provides a brief overview of the available AssetGroupBridge methods, along with available REST Calls. For a full description of AssetGroupBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
create	Creates a new Asset Group.	PUT <server>/services/v2/rest/assetGroup **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/assetGroup
delete	Deletes an Asset Group.	DELETE <server>/services/v2/rest/assetGroup/{id}
find (by criteria)	Finds Asset Groups based on search criteria.	POST <server>/services/v2/rest/assetGroup/find
find (by IDs)	Finds Asset Groups based on a list of identifiers.	POST <server>/services/v2/rest/assetGroup/findByIds
find (by alternate ID)	Finds an Asset Group based on the alternate identifier.	GET <server>/services/v2/rest/assetGroup/{id}
findById	Finds an Asset Group based on its platform identifier.	GET <server>/services/v2/rest/assetGroup/{id}
findOne	Returns the first Asset Group found that meets specified search criteria.	POST <server>/services/v2/rest/assetGroup/fin

		dOne
getAssets	Returns the Assets that are members of a specified Asset Group.	GET <server>/services/v2/rest/assetGroup/assets
getChildren	Returns the child Asset Groups of a specified Asset Group.	GET <server>/services/v2/rest/assetGroup/children
toString	Generates a string representing a specified Asset Group.	N/A
update	Updates an existing Asset Group.	POST <server>/services/v2/rest/assetGroup/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations:

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/assetGroup/bulk/create
bulkDelete	POST <server>/services/v2/rest/assetGroup/bulk/delete
bulkSave	POST <server>/services/v2/rest/assetGroup/bulk/save
bulkUpdate	POST <server>/services/v2/rest/assetGroup/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetGroupBridge.html>

Wednesday, March 01, 2017 10:25 AM

Create an AssetGroup

```
1 def myAssetGroup = new AssetGroup(name:"My Asset Group")
2 assetGroupBridge.create(myAssetGroup)
```

Find an AssetGroup by Criteria

```
1 FindAssetGroupResult existingAssetGroupsResult = assetGroupBridge.find(new
  AssetGroupCriteria())
```

Delete an AssetGroup

```
1 assetGroupBridge.delete(myAssetGroupFound)
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetGroupBridge.html>

Create an AssetGroup

Endpoint	/services/v2/rest/assetGroup
Verb(s)	PUT
Input Object	AssetGroup
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AssetGroup xmlns="http://www.axeda.com/services/v2">
  <name>groupName1367256555450</name>
  <description>groupDescription1367256555450</description>

</AssetGroup>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>/Root Asset
Group/groupName1367256555450</v2:ref>
      <v2:id>853</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find an AssetGroup by Criteria

Endpoint	/services/v2/rest/assetGroup/find
Verb(s)	POST
Input Object	AssetGroupCriteria
Output Object	FindAssetGroupResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AssetGroupCriteria xmlns="http://www.axeda.com/services/v2">

  <name>groupName1367590868365*</name>
</AssetGroupCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindAssetGroupResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
totalCount="2">
  <v2:criteria>
    <v2:name>groupName1367590868365*</v2:name>
  </v2:criteria>
  <v2:assetGroups>
    <v2:assetGroup xsi:type="v2:AssetGroup" id="/Root Asset
Group/groupName13675908683651" systemId="134"
label="groupName13675908683651" detail="/Root Asset Group"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/assetGroup/134">
      <v2:name>groupName13675908683651</v2:name>
      <v2:parentAssetGroup id="/Root Asset Group" systemId="1"
label="Root Asset Group" detail="" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/assetGroup/1"/>

      <v2:description>
groupDescription13675908683651</v2:description>
      <v2:parentGroupHierarchy>/Root Asset
Group/groupName13675908683651</v2:parentGroupHierarchy>
    </v2:assetGroup>
    <v2:assetGroup xsi:type="v2:AssetGroup" id="/Root Asset
Group/groupName13675908683652" systemId="135"
label="groupName13675908683652" detail="/Root Asset Group"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/assetGroup/135">
      <v2:name>groupName13675908683652</v2:name>
      <v2:parentAssetGroup id="/Root Asset Group" systemId="1"
label="Root Asset Group" detail="" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/assetGroup/1"/>

      <v2:description>
groupDescription13675908683652</v2:description>
      <v2:parentGroupHierarchy>/Root Asset
Group/groupName13675908683652</v2:parentGroupHierarchy>
    </v2:assetGroup>
  </v2:assetGroups>
</v2:FindAssetGroupResult>
```

Update an AssetGroup

Endpoint	/services/v2/rest/assetGroup/854
Verb(s)	POST
Input Object	AssetGroup
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AssetGroup xmlns="http://www.axeda.com/services/v2"
systemId="854">
  <name>changed_1367256558542</name>
  <description>Changed
groupDescription1367256558542</description>
</AssetGroup>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
```

```
    <v2:succeeded>
      <v2:success>
        <v2:ref>854</v2:ref>
        <v2:id>854</v2:id>
      </v2:success>
    </v2:succeeded>
    <v2:failures/>
  </v2:ExecutionResult>
```

Save an AssetGroup

Endpoint	/services/v2/rest/assetGroup
Verb(s)	POST
Input Object	AssetGroup
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<AssetGroup xmlns="http://www.axeda.com/services/v2"
systemId="855">
  <name>changed_1367256560558</name>
  <description>Changed
groupDescription1367256560558</description>
</AssetGroup>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>855</v2:ref>
      <v2:id>855</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete an AssetGroup

Endpoint	/services/v2/rest/assetGroup/856
Verb(s)	
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>856</v2:ref>
      <v2:id>856</v2:id>
```

```
10      </v2:success>
      </v2:succeeded>
      <v2:failures/>
    </v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/AssetGroupBridge.html>

CustomObjectBridge

A Custom Object provides a mechanism for customizing rules and actions for use on the Axeda Platform. Based on the Groovy scripting language, the Custom Object feature provides a simple way to write custom Java code for the Axeda Platform.

Note: You can also use the ScriptoService to run scripts defined in the Axeda Platform. For more information about the ScriptoService, refer to the *Axeda® Platform Web Services Developer's Reference, v2REST*.

Custom Objects can be created and executed using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and executing Custom Objects through the API using the CustomObjectBridge.

The CustomObjectBridge provides Create, Read/Find, Update, and Delete (CRUD) functions, as well as invocation operations, for Custom Objects.

Synchronous and Asynchronous Execution

Custom Objects can be executed both synchronously and asynchronously:

Synchronous execution — A Custom Object executes a Groovy script and waits for its output.

Asynchronous execution — A Custom Object executes a Groovy script, but does not wait for its output. This is sometimes referred to as “fire and forget” execution. Optionally, an asynchronous Custom Object execution can also periodically check the status of an initiated task, and retrieve the results of a task. It is also possible to register a “callback” Custom Object which will be invoked with the output of the first ansynchronously-invoked Custom Object.

For more details about asynchronous execution of Custom Objects, see Asynchronous Custom Object Execution.

Using Call and Request to Access Custom Object Parameters

The Call Object

The Call object holds the various information available when a Custom Object is invoked. A Groovy script can access this information by importing these static methods.

For example, if you did the following:

```
Bridges.customObjectBridge.execute("myscript", [foo: "bar"]);
```

Or, using the ScriptoService:

```
curl -X POST -d "foo=bar" -H "Content-Type: text/plain"
"http://host/services/v1/rest/Scripto/execute/myscript"
```

Then in a Groovy script, you could access the foo=bar parameters as follows:

```
import static com.axeda.drm.services.customobject.Call.*;
def fooParam = parameters.foo // should be "bar"
```

You could also access the current username:

```
def user = username // should be e.g., "admin"
```

For a full description of Call object methods, see the Axeda API Specification.

The Request Object

The Request object is an extenstion of the Call object that holds the various HTTP request information that is posted to Scripto.

The com.axeda.drm.sdk.scripto.Request object stores a specified subset of HTTP request information that is relayed to Scripto. The Scripto service extracts the information from a Scripto request and stores it in the Request object. When a particular Groovy script is executed, you can reference the static methods on the Request to access HTTP request information.

For example, if you did the following:

```
curl -X POST -d "foo=bar" -H "Content-Type: text/plain" "http://host/services/
v1/rest/Scripto/execute/MyScript"
```

Then in the Groovy code, you could extract the foo=bar parameters as follows:

```
import static com.axeda.drm.sdk.scripto.Request.*;
def fooParam = parameters.foo // should be "bar"
```

You could also extract the Content-Type header:

```
def contentType = Request.headers.'Content-Type' // should be "text/plain"
```

Note: For more information, refer to the API Specification.

Custom Object Permissions

Access control to Custom Objects is governed by privileges. Privileges can be applied globally to all Custom Objects, or to an individual Custom Object.

Individual Custom Object permissions can be added using the MappedPrivilege object. A Mapped Privilege represents an association between a privilege (permission) and a User Group. Each Custom Object can have a list of these Mapped Privileges.

The individual Privileges that can be applied to a Custom Object are Read, Write, and Execute. At the individual level, setting the Delete permission will have no affect. Instead, any user with the Write permission will be allowed to delete the Custom Object. By default, Custom Objects have no individual permissions applied.

Access to an individual Custom Object can be restricted based on permissions. If at least one User Group is

assigned one of the Read/Write/Execute permissions for a specific Custom Object, access to that Custom Object will be restricted to that User Group (or any other User Groups that have been explicitly granted that permission for that Custom Object). Access is restricted based on specific permissions, so if only the Write permission is restricted, the access restriction will only apply to Write. If no User Group is assigned a particular permission, that type of access will not be restricted for the Custom Object, but access can still be restricted at the global level.

All users have global Execute permission by default. In order to perform Read and Write operations on a Custom Object, a user needs those global access privileges, or belong to a User Group with Read and Write permissions for the Custom Object. Access privileges for an individual Custom Object (assigned via User Groups) takes precedence over global access settings. For example, if Read access has been restricted for a Custom Object through a User Group, users with global Read permission will not be able to access that object unless they belong to that User Group.

The following table lists the Read, Write, and Execute (RWX) permissions for a Custom Object based on global and User Group permission settings. Note: in the scenario represented by the highlighted row, users who do not belong to the group should not have Write access, but currently they do because the global Write permission is enabled.

Global CO Read	Global CO Write	CO Read	CO Write	CO Execute	People Not in This Group	People In This Group	Notes
					X	X	only admin can update
			x		X	RWX	
		x			X	RX	only admin can update
		x	x		X	RWX	
	x				RWX	RWX	
	x		x		X	RWX	
	x	x			WX	RWX	
	x	x	x		X	RWX	
x					RX	RX	
x			x		RX	RWX	
x		x			RX	RX	only admin can update
x		x	x		RX	RWX	
x	x				RWX	RWX	
x	x		x		RX	RWX	
x	x	x			RWX	RWX	WRITE IS NOT RESTRICTED thus you get the read
x	x	x	x		X	RWX	
				x	-	X	only admin can update
			x	x	-	RWX	
		x		x	-	RX	only admin can update
		x	x	x	-	RWX	
	x			x	RW	RWX	
	x		x	x	-	RWX	
	x	x		x	-	RWX	
	x	x	x	x	-	RWX	
x				x	R	RX	only admin can update
x			x	x	R	RWX	
x		x		x	R	RX	only admin can update
x		x	x	x	R	RWX	
x	x			x	RW	RWX	
x	x		x	x	R	RWX	
x	x	x		x	RW	RWX	
x	x	x	x	x	-	RWX	

Methods

The following table provides a brief overview of the available CustomObjectBridge methods, along with available REST calls. For a full description of CustomObjectBridge methods, see the Axeda API Specification.

Method	Description	REST Call
cancelAsync	Attempts to cancel the asynchronous execution of a Custom	N/A

(by name)	Object based on name.	
cancelAsync (by UUID)	Attempts to cancel the asynchronous execution of a Custom Object based on UUID.	N/A
compile	Checks to see if the source of a Custom Object compiles.	POST <server>/services/v2/rest/customObject/compile
create	Creates a Custom Object.	PUT <server>/services/v2/rest/customObject **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/customObject
delete	Deletes a Custom Object.	DELETE <server>/services/v2/rest/customObject/id/{id}
deleteAsyncResult (by name)	Attempts to delete the execution result of an asynchronous CustomObject based on name.	N/A
deleteAsyncResult (by UUID)	Attempts to delete the execution result of an asynchronous CustomObject based on UUID.	N/A
execute	Executes a Custom Object.	N/A
find (by criteria)	Finds Custom Objects based on search criteria.	POST <server>/services/v2/rest/customObject/find
find (by IDs)	Finds Custom Objects based on a list of IDs.	POST <server>/services/v2/rest/customObject/findByIds
find (by alternate ID)	Finds a Custom Objects based on its alternate identifier.	GET <server>/services/v2/rest/customObject/id/{id}
findById	Finds a Custom Object based on its platform identifier.	GET <server>/services/v2/rest/customObject/id/{id}
findOne	Returns the first Custom Object found that meets specified search criteria.	POST <server>/services/v2/rest/customObject/findOne
getAsyncResult	Retrieves the result of a previous request to initiate asynchronous execution of a Custom Object .	N/A
getAsyncStatus	Retrieves the status of a previous request to initiate asynchronous execution of a Custom Object.	N/A
initiateAsync	Initiates the asynchronous execution of a Custom Object.	N/A
InitiateAsyncWithCallback	Initiates the execution of a CustomObject, and the execution of a callback Custom Object after the target Custom Object execution is complete.	N/A
toString	Generates a string representing a specified Custom Object.	N/A
update	Updates an existing Custom Object.	POST

		<server>/services/v2/rest/customObject/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using Custom Object Collections.

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/customObject/bulk/create
bulkDelete	POST <server>/services/v2/rest/customObject/bulk/delete
bulkSave	POST <server>/services/v2/rest/customObject/bulk/save
bulkUpdate	POST <server>/services/v2/rest/customObject/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/CustomObjectBridge.html>

Create a CustomObject

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.CustomObjectCriteria
import com.axeda.services.v2.CustomObjectType
import com.axeda.services.v2.CustomObject

// CREATE
CustomObject customObject = new CustomObject([name: "CreatedScript", type:
CustomObjectType.ACTION,
source:
"""  // ***begin script***
import com.axeda.drm.sdk.customobject.Call
import net.sf.json.JSONObject

//
// CreatedScript.groovy
//
// This script was created by another script.
//
//

logger.info(Call.parameters.param1)
logger.info(Call.parameters.param2)

def response = [
      param1: Call.parameters.param1, param2: Call.parameters.param2
  ]

return ['Content-Type': 'application/text', 'Content': response]

"""  // ***end script***
])
customObjectBridge.create(customObject)
result = customObjectBridge.execute(customObject.name, [param1: "Hello",
param2:"World"])
```

Find a CustomObject by Criteria

```
CustomObjectCriteria customObjectCriteria = new CustomObjectCriteria()

customObjectCriteria.setName "CreatedScript"
def co = customObjectBridge.find customObjectCriteria
result = customObjectBridge.execute(co.customObjects[0].label, [param1:
"Hello", param2:"World"])
assert result.Content == [param1:"Hello", param2:"World"]
```

Update a CustomObject

```
def customObj = customObjectBridge.findById(co.customObjects[0].id)
customObj.description = "Custom Description"
customObjectBridge.update(customObj)
customObj = customObjectBridge.find([co.customObjects[0].id])
assert customObj.description[0] == "Custom Description"
```

Delete a CustomObject

```
def result = customObjectBridge.delete(customObj)
// returns an Execution Result
assert result.successful == true
// the 'succeeded' property is a list of Successful Operation objects

assert result.succeeded[0].class.name ==
"com.axeda.services.v2.SuccessfulOperation"
```

Asynchronous CustomObject

```
//This is a "fire and forget" chain in which one Groovy script can execute

//another Groovy script. Note that the first script does not wait for the

//output of the invoked Groovy script.


 import static com.axeda.sdk.v2.dsl.Bridges.*

 // last parameter is a timeout for the execution measured in seconds

 customObjectBridge.initiateAsync("CreatedScript", [param1:"Fred", param2:
"35"], 600)

 // asynchronous with Callback

 // returns an Execution Result
 def result =
customObjectBridge.initiateAsyncWithCallback("CreatedScript",
[param1:"Fred", param2: "35"], "Script2", 600)
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/CustomObjectBridge.html>

Create a CustomObject

Endpoint	/services/v2/rest/customObject
Verb(s)	PUT
Input Object	CustomObject
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://www.axeda.com/services/v2">
  <name>name1367258084278</name>
  <type>ACTION</type>
  <params>
    <customObjectParam>
      <variableName>var11367258084278</variableName>
      <displayName>displayName11367258084278</displayName>

    </customObjectParam>
    <customObjectParam>
      <variableName>var21367258084278</variableName>
      <displayName>displayName21367258084278</displayName>

    </customObjectParam>
  </params>
  <description>description1367258084278</description>
  <source>import com.axeda.*</source>
  <languageType>GROOVY</languageType>
</CustomObject>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>name1367258084278</v2:ref>
      <v2:id>70</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find a CustomObject by Criteria

Endpoint	/services/v2/rest/customObject/find
Verb(s)	POST
Input Object	CustomObjectCriteria

Output Object FindCustomObjectResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectCriteria
xmlns="http://www.axeda.com/services/v2" pageSize="10"
pageNumber="1">
  <name>*1367590980592*</name>
  <exactName>false</exactName>
</CustomObjectCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindCustomObjectResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="3">
  <v2:criteria pageSize="10" pageNumber="1">
    <v2:name>*1367590980592*</v2:name>
    <v2:exactName>false</v2:exactName>
    <v2:types/>
  </v2:criteria>
  <v2:customObjects>
    <v2:customObject xsi:type="v2:CustomObject"
id="name1367590980592a" systemId="37"
label="name1367590980592a" detail="action"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/customObject/37">
      <v2:name>name1367590980592a</v2:name>
      <v2:type>ACTION</v2:type>
      <v2:params>
        <v2:customObjectParam>
          <v2:variableName>
var11367590980592a</v2:variableName>
          <v2:displayName>
displayName11367590980592a</v2:displayName>
        </v2:customObjectParam>
        <v2:customObjectParam>
          <v2:variableName>
var21367590980592a</v2:variableName>
          <v2:displayName>
displayName21367590980592a</v2:displayName>
        </v2:customObjectParam>
      </v2:params>
      <v2:description>
description1367590980592a</v2:description>
      <v2:source>import com.axeda.*
</v2:source>
      <v2:languageType>GROOVY</v2:languageType>
      <v2:systemObject>false</v2:systemObject>
      <v2:mappedPrivileges/>
    </v2:customObject>
    <v2:customObject xsi:type="v2:CustomObject"
id="name1367590980592b" systemId="38"
label="name1367590980592b" detail="action"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/customObject/38">
      <v2:name>name1367590980592b</v2:name>
      <v2:type>ACTION</v2:type>
      <v2:params>
        <v2:customObjectParam>
          <v2:variableName>
var11367590980592b</v2:variableName>
```

```
          <v2:displayName>
displayName11367590980592b</v2:displayName>
        </v2:customObjectParam>
        <v2:customObjectParam>
          <v2:variableName>
var21367590980592b</v2:variableName>
          <v2:displayName>
displayName21367590980592b</v2:displayName>
        </v2:customObjectParam>
      </v2:params>
      <v2:description>
description1367590980592b</v2:description>
      <v2:source>import com.axeda.*
</v2:source>
      <v2:languageType>GROOVY</v2:languageType>
      <v2:systemObject>false</v2:systemObject>
      <v2:mappedPrivileges/>
    </v2:customObject>
    <v2:customObject xsi:type="v2:CustomObject"
id="name1367590980592c" systemId="39"
label="name1367590980592c" detail="action"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/customObject/39">
      <v2:name>name1367590980592c</v2:name>
      <v2:type>ACTION</v2:type>
      <v2:params>
        <v2:customObjectParam>
          <v2:variableName>
var11367590980592c</v2:variableName>
          <v2:displayName>
displayName11367590980592c</v2:displayName>
        </v2:customObjectParam>
        <v2:customObjectParam>
          <v2:variableName>
var21367590980592c</v2:variableName>
          <v2:displayName>
displayName21367590980592c</v2:displayName>
        </v2:customObjectParam>
      </v2:params>
      <v2:description>
description1367590980592c</v2:description>
      <v2:source>import com.axeda.*
</v2:source>
      <v2:languageType>GROOVY</v2:languageType>
      <v2:systemObject>false</v2:systemObject>
      <v2:mappedPrivileges/>
    </v2:customObject>
  </v2:customObjects>
</v2:FindCustomObjectResult>
```

Update a CustomObject

Endpoint	/services/v2/rest/customObject/id/69
Verb(s)	POST
Input Object	CustomObject
Output Object	ExecutionResult
XML Request	

```
1 <?xml version="1.0" encoding="UTF-8"?>
```

```
<CustomObject xmlns="http://www.axeda.com/services/v2"
systemId="69">
  <name>name1367258083505</name>
  <type>ACTION</type>
  <params>
    <customObjectParam>
      <variableName>var11367258083505</variableName>
      <displayName>displayName11367258083505</displayName>

    </customObjectParam>
    <customObjectParam>
      <variableName>var21367258083505</variableName>
      <displayName>displayName21367258083505</displayName>

    </customObjectParam>
  </params>
  <description>description1367258083505</description>
  <source>import com.axeda.*</source>
  <languageType>GROOVY</languageType>
</CustomObject>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>69</v2:ref>
      <v2:id>69</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find One CustomObject

Endpoint	/services/v2/rest/customObject/findOne
Verb(s)	POST
Input Object	CustomObjectCriteria
Output Object	CustomObject

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectCriteria xmlns="http://www.axeda.com/services/v2">

  <name>*1367590995799*</name>
</CustomObjectCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:CustomObject xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="name1367590995799a" systemId="59" label="name1367590995799a"
detail="action" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/customObject/59">

```

```
 8
 9    <v2:name>name1367590995799a</v2:name>
10    <v2:type>ACTION</v2:type>
11    <v2:params>
12      <v2:customObjectParam>
13        <v2:variableName>var11367590995799a</v2:variableName>
14
15        <v2:displayName>
16  displayName11367590995799a</v2:displayName>
17      </v2:customObjectParam>
18      <v2:customObjectParam>
19        <v2:variableName>var21367590995799a</v2:variableName>
20
21        <v2:displayName>
    displayName21367590995799a</v2:displayName>
      </v2:customObjectParam>
    </v2:params>
    <v2:description>description1367590995799a</v2:description>

    <v2:source>import com.axeda.*
  </v2:source>
    <v2:languageType>GROOVY</v2:languageType>
    <v2:systemObject>false</v2:systemObject>
    <v2:mappedPrivileges/>
  </v2:CustomObject>
```

Delete a CustomObject

Endpoint	/services/v2/rest/customObject/id/71
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <v2:ExecutionResult xmlns:v2
 3  ="http://www.axeda.com/services/v2"
 4  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 5  successful="true" totalCount="1">
 6    <v2:succeeded>
 7      <v2:success>
 8        <v2:ref>71</v2:ref>
 9        <v2:id>71</v2:id>
10      </v2:success>
      </v2:succeeded>
      <v2:failures/>
    </v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/CustomObjectBridge.html>

DataItemBridge

A Data Item is a named reading, such as a sensor reading, the current location of a vehicle (latitude, longitude, and altitude coordinates), or the available memory on a device.

On the Axeda Platform, a Data Item Definition assigns a name, a Model association, and a set of properties for an Data Item. Data Item definitions are created and managed at the Model level.

A Data Item Value holds the value of a Data Item on a specific Data Item at a specific time. In the Axeda Platform API, Data Item Values are represented by the DataItemValue object.

Data Item Values -- such as hourly temperatures, odometer readings, or daily usage statistics -- are time-stamped in a sequence. Data Item Values can be used on the Axeda Platform in Expression Rules or Custom Objects that perform business logic.

Data Items can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating Data Items through the API using the DataItemBridge object.

The DataItemBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for DataItem objects. You can also use the DataItemBridge to get source and target Data Items.

Methods

The following table provides a brief overview of the DataItemBridge methods, along with available REST calls. For a full description of DataItemBridge methods, see the Axeda API Specification.

Method	Description	REST Call
create	Creates a new Data Item.	PUT <server>/services/v2/rest/dataItem **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/dataItem
delete	Deletes a Data Item.	DELETE <server>/services/v2/rest/dataItem/id/{id}
find (by criteria)	Finds Data Items based on search criteria.	POST <server>/services/v2/rest/dataItem/find
find (by IDs)	Finds Data Items based on a list of identifiers.	POST <server>/services/v2/rest/dataItem/findByIds
find (by alternate	Finds a Data Item based on the alternate identifier.	GET <server>/services/v2/rest/dataItem/id/{

ID)		id}
findById	Finds an Data Item based on its platform identifier.	GET <server>/services/v2/rest/dataItem/id/{id}
findCurrentValues	Returns the current values of the specified Data Items.	POST <server>/services/v2/rest/dataItem/findCurrentValues
findHistoricalValues	Returns the historical values of the specified Data Items.	POST <server>/services/v2/rest/dataItem/findHistoricalValues
findOne	Returns the first Data Item found that meets specified search criteria.	POST <server>/services/v2/rest/dataItem/findOne
getSourceDataItems	Returns the source Data Items associated with the specified target Data Item.	GET <server>/services/v2/rest/dataItem/sourceDataItems
getTargetDataItems	Returns the target Data Items associated with the specified source Data Item.	GET <server>/services/v2/rest/dataItem/targetDataItems
toString	Generates a string representing a specified Data Item.	N/A
update	Updates an existing Data Item.	POST <server>/services/v2/rest/dataItem/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using Data Item Collections.

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/dataItem/bulk/create
bulkDelete	POST <server>/services/v2/rest/dataItem/bulk/delete
bulkSave	POST <server>/services/v2/rest/dataItem/bulk/save
bulkUpdate	POST <server>/services/v2/rest/dataItem/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/DataItemBridge.html>

Create a DataItem

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def dataItem = new DataItem(
    model: new ModelReference(id: "Vending Machine 5000"),
    name: "Temperature",
    alias: "temp",
    type: DataItemType.ANALOG
)

def result = dataItemBridge.create(dataItem)
if (result.successful) {
  println "The data item was created"
}
else {
  assert result.failures.any() { it.code == "NOT_UNIQUE" }
}
```

Delete a DataItem

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def dataItem = new DataItem(id: "Vending Machine 5000||External
temperature level")

def result = dataItemBridge.delete(dataItem)
if (result.successful) {
  println "The data item was deleted"
}
else {
  assert result.failures.every { it.code == "OBJECT_NOT_FOUND" }
}
```

Find a DataItem by Alternate ID

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def foundDataItem = dataItemBridge.find("Vending Machine 5000||Stock
level")
assert foundDataItem
```

Find a DataItem by Criteria

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def model = modelBridge.find("Vending Machine 5000")
assert model
```

```
6
7 def findResult = dataItemBridge.find(
8     new DataItemCriteria(modelId: model.systemId, name: "* level", types:
9 [DataItemType.ANALOG]))
  assert !findResult.dataItems.isEmpty()
```

Find a DataItem by System ID

```
1 import com.axeda.drm.sdk.customobject.*
2 import static com.axeda.sdk.v2.dsl.Bridges.*
3
4 String dataItemId = Call.parameters.dataItemId
5
6 def foundDataItem = dataItemBridge.findById(dataItemId)
7 assert foundDataItem
```

Find Current DataItem Values

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.services.v2.*
3
4 def asset = assetBridge.find("Vending Machine 5000||A36")
5 assert asset
6
7 def findResult = dataItemBridge.findCurrentValues(
8     new CurrentDataItemValueCriteria(
9         assetId: asset.systemId,
10        name: "* level",
11        types: [DataItemType.ANALOG]))
12 findResult.dataItemValues.each { println("Value for ${it.dataItem.label}:
  ${it.value}") }s
```

Find Historical DataItem Values

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.services.v2.*
3
4 def asset = assetBridge.find("Vending Machine 5000||A36")
5 assert asset
6
7 def findResult = dataItemBridge.findHistoricalValues(
8     new HistoricalDataItemValueCriteria(
9         assetId: asset.systemId,
10        startDate: new Date() - 1,
11        endDate: new Date()))
12 assert findResult
13
14 findResult.dataItemValues.each { println("Historical value for
  ${it.dataItem.label}: ${it.value}") }
```

Find One DataItem

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.services.v2.*
3
4 def model = modelBridge.find("Vending Machine 5000")
5 assert model
```

```
6
7 def findResult = dataItemBridge.findOne(
8     new DataItemCriteria(modelId: model.systemId, name: "* level", types:
9 [DataItemType.ANALOG]))
  assert findResult
```

Get Source DataItems

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.services.v2.*
3
4 def sourceItems = dataItemBridge.getSourceDataItems(new
5 DataItemReference(id: "Vending Machine 5000||Stock level"))
  sourceItems.each { println("Found source item: ${it}") }
```

Get Target DataItems

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.services.v2.*
3
4 def targetItems = dataItemBridge.getTargetDataItems(new
5 DataItemReference(id: "Vending Machine 5000||Stock level"))
6 targetItems.each { println("Found target item: ${it}") }
7
8 def sourceItems = dataItemBridge.getSourceDataItems(new
  DataItemReference(id: "Vending Machine 5000||Stock level"))
  sourceItems.each { println("Found source item: ${it}") }
```

Save a DataItem

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.services.v2.*
3
4 def dataItem = new DataItem(
5     id: "Vending Machine 5000||Stock level",
6     name: "Stock level",
7     model: new ModelReference(id: "Vending Machine 5000"),
8     alias: "stock_level",
9     forwarded: true,
10    type: DataItemType.ANALOG
11 )
12
13 def result = dataItemBridge.save(dataItem)
14 assert result.successful
```

Update a DataItem

```
1 import static com.axeda.sdk.v2.dsl.Bridges.*
2 import com.axeda.services.v2.*
3
4 def dataItem = new DataItem(
5     id: "Vending Machine 5000||Stock level",
6     name: "Stock level",
7     model: new ModelReference(id: "Vending Machine 5000"),
8     alias: "stock_level",
9     visible: "false",
10    type: DataItemType.ANALOG
```

```
)

def result = dataItemBridge.update(dataItem)
assert result.successful
```

Bulk Create

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def dataItem1 = new DataItem(
    model: new ModelReference(id: "Vending Machine 5000"),
    name: "Power level",
    alias: "stock",
    type: DataItemType.ANALOG
)

def dataItem2 = new DataItem(
    model: new ModelReference(id: "Vending Machine 5000"),
    name: "Bank level",
    alias: "bank",
    type: DataItemType.ANALOG
)

def result = dataItemBridge.create([dataItem1, dataItem2])
if (result.successful) {
  println "The data items were created"
}
else {
  assert result.failures.every { it.code == "NOT_UNIQUE" }
}
```

Bulk Delete

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def dataItem1 = new DataItem(
        model: new ModelReference(id: "Vending Machine 5000"),
        name: "Power level",
        alias: "stock",
        type: DataItemType.ANALOG
)

def dataItem2 = new DataItem(
        model: new ModelReference(id: "Vending Machine 5000"),
        name: "Bank level",
        alias: "bank",
        type: DataItemType.ANALOG
)

def result = dataItemBridge.create([dataItem1, dataItem2])
if (result.successful) {
    println "The data items were created"
}
else {
    assert result.failures.every { it.code == "NOT_UNIQUE" }
}
import static com.axeda.sdk.v2.dsl.Bridges.*
```

```
import com.axeda.services.v2.*

def dataItem1 = new DataItem(id: "Vending Machine 5000||Humidity level")

def dataItem2 = new DataItem(id: "Vending Machine 5000||Jammed")

def result = dataItemBridge.delete([dataItem1, dataItem2])
if (result.successful) {
  println "The data items were deleted"
}
else {
  assert result.failures.every { it.code == "OBJECT_NOT_FOUND" }
}
```

Bulk Save

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def dataItem1 = new DataItem(
    id: "Vending Machine 5000||Stock level",
    name: "Stock level",
    model: new ModelReference(id: "Vending Machine 5000"),
    alias: "stock_level",
    visible: "false",
    type: DataItemType.ANALOG
)

def dataItem2 = new DataItem(
    model: new ModelReference(id: "Vending Machine 5000"),
    name: "Additional info",
    type: DataItemType.STRING
)

def result = dataItemBridge.save([dataItem1, dataItem2])
if (result.successful) {
  println "One data item created and one was updated"
}
else {
  assert result.failures.every() { it.code == "NOT_UNIQUE" }
}
```

Bulk Update

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def dataItem1 = new DataItem(
    id: "Vending Machine 5000||Stock level",
    name: "Stock level",
    model: new ModelReference(id: "Vending Machine 5000"),
    alias: "stock_level",
    visible: "false",
    type: DataItemType.ANALOG
)

def dataItem2 = new DataItem(
    id: "Vending Machine 5000||Change level",
    name: "Change level",
```

```
    model: new ModelReference(id: "Vending Machine 5000"),
    alias: "change_level",
    visible: "false",
    type: DataItemType.ANALOG
)

def result = dataItemBridge.update([dataItem1, dataItem2])
assert result.successful
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/DataItemBridge.html>

Create a DataItem

Endpoint	/services/v2/rest/dataItem
Verb(s)	PUT
Input Object	DataItem
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItem xmlns="http://www.axeda.com/services/v2">
  <name>item1367258892180</name>
  <alias>alias1367258892180</alias>
  <description>item description</description>
  <model systemId="818"/>
  <type>STRING</type>
  <readOnly>false</readOnly>
  <visible>true</visible>
  <forwarded>false</forwarded>
  <storage>STORED</storage>
  <stringMinLength>1</stringMinLength>
  <stringMaxLength>100</stringMaxLength>
  <analogRangeLowerLimit>1.0</analogRangeLowerLimit>
  <analogRangeUpperLimit>3.0</analogRangeUpperLimit>
</DataItem>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>model no 13672588827090
||item1367258892180</v2:ref>
      <v2:id>52</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Update a DataItem

Endpoint	/services/v2/rest/dataItem/id/53
Verb(s)	POST
Input Object	DataItem
Output Object	ExecutionResult

XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

Request

```
<DataItem xmlns="http://www.axeda.com/services/v2"
systemId="53">
  <name>changeditem1367258893697</name>
  <alias>changedalias1367258893697</alias>
  <description>changeditem description</description>
  <model id="model no 13672588827090" systemId="818"
label="standalone" detail="model no 13672588827090"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/818"/>
  <type>STRING</type>
  <readOnly>true</readOnly>
  <visible>false</visible>
  <forwarded>true</forwarded>
  <storage>STORED</storage>
  <stringMinLength>2</stringMinLength>
  <stringMaxLength>99</stringMaxLength>
  <analogRangeLowerLimit>2.0</analogRangeLowerLimit>
  <analogRangeUpperLimit>2.0</analogRangeUpperLimit>
</DataItem>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>53</v2:ref>
      <v2:id>53</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Save a DataItem

Endpoint	/services/v2/rest/dataItem
Verb(s)	POST
Input Object	DataItem
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItem xmlns="http://www.axeda.com/services/v2"
systemId="54">
  <name>changeditem1367258895133</name>
  <alias>changedalias1367258895133</alias>
  <description>changeditem description</description>
  <model id="model no 13672588827090" systemId="818"
label="standalone" detail="model no 13672588827090"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/818"/>
  <type>STRING</type>
  <readOnly>true</readOnly>
  <visible>false</visible>
  <forwarded>true</forwarded>
  <storage>STORED</storage>
```

```
    <stringMinLength>2</stringMinLength>
    <stringMaxLength>99</stringMaxLength>
    <analogRangeLowerLimit>2.0</analogRangeLowerLimit>
    <analogRangeUpperLimit>2.0</analogRangeUpperLimit>
  </DataItem>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>54</v2:ref>
      <v2:id>54</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete a DataItem

Endpoint	/services/v2/rest/dataItem/id/55
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>55</v2:ref>
      <v2:id>55</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find a DataItem by Criteria

Endpoint	/services/v2/rest/dataItem/find
Verb(s)	POST
Input Object	DataItemCriteria
Output Object	FindDataItemResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<DataItemCriteria xmlns="http://www.axeda.com/services/v2">

  <name>item*</name>
  <modelId>80</modelId>
```

```
  <types>
    <dataItemType>STRING</dataItemType>
  </types>
</DataItemCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindDataItemResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="3">
  <v2:criteria>
    <v2:name>item*</v2:name>
    <v2:modelId>80</v2:modelId>
    <v2:types>
      <v2:dataItemType>STRING</v2:dataItemType>
    </v2:types>
  </v2:criteria>
  <v2:dataItems>
    <v2:dataItem xsi:type="v2:DataItem" id="model no
13675910255310||item13675910255250" systemId="40"
label="item13675910255250" detail="STRING"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/dataItem/40">
      <v2:name>item13675910255250</v2:name>
      <v2:alias>alias13675910255250</v2:alias>
      <v2:description>item description</v2:description>
      <v2:model id="model no 13675910255310" systemId="80"
label="standalone" detail="model no 13675910255310"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/80"/>
      <v2:type>STRING</v2:type>
      <v2:readOnly>false</v2:readOnly>
      <v2:visible>true</v2:visible>
      <v2:forwarded>false</v2:forwarded>
      <v2:storage>STORED</v2:storage>
      <v2:stringMinLength>1</v2:stringMinLength>
      <v2:stringMaxLength>100</v2:stringMaxLength>
      <v2:analogRangeLowerLimit>1.0</v2:analogRangeLowerLimit>

      <v2:analogRangeUpperLimit>3.0</v2:analogRangeUpperLimit>

    </v2:dataItem>
    <v2:dataItem xsi:type="v2:DataItem" id="model no
13675910255310||item13675910255251" systemId="41"
label="item13675910255251" detail="STRING"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/dataItem/41">
      <v2:name>item13675910255251</v2:name>
      <v2:alias>alias13675910255251</v2:alias>
      <v2:description>item description</v2:description>
      <v2:model id="model no 13675910255310" systemId="80"
label="standalone" detail="model no 13675910255310"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/80"/>
      <v2:type>STRING</v2:type>
      <v2:readOnly>false</v2:readOnly>
      <v2:visible>true</v2:visible>
      <v2:forwarded>false</v2:forwarded>
      <v2:storage>STORED</v2:storage>
      <v2:stringMinLength>1</v2:stringMinLength>
      <v2:stringMaxLength>100</v2:stringMaxLength>
```

```
57        <v2:analogRangeLowerLimit>1.0</v2:analogRangeLowerLimit>

          <v2:analogRangeUpperLimit>3.0</v2:analogRangeUpperLimit>

        </v2:dataItem>
        <v2:dataItem xsi:type="v2:DataItem" id="model no
13675910255310||item13675910255252" systemId="42"
label="item13675910255252" detail="STRING"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/dataItem/42">
          <v2:name>item13675910255252</v2:name>
          <v2:alias>alias13675910255252</v2:alias>
          <v2:description>item description</v2:description>
          <v2:model id="model no 13675910255310" systemId="80"
label="standalone" detail="model no 13675910255310"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/80"/>
          <v2:type>STRING</v2:type>
          <v2:readOnly>false</v2:readOnly>
          <v2:visible>true</v2:visible>
          <v2:forwarded>false</v2:forwarded>
          <v2:storage>STORED</v2:storage>
          <v2:stringMinLength>1</v2:stringMinLength>
          <v2:stringMaxLength>100</v2:stringMaxLength>
          <v2:analogRangeLowerLimit>1.0</v2:analogRangeLowerLimit>

          <v2:analogRangeUpperLimit>3.0</v2:analogRangeUpperLimit>

        </v2:dataItem>
      </v2:dataItems>
    </v2:FindDataItemResult>
```

Find a DataItem by Alternate ID

Endpoint	/services/v2/rest/dataItem/model%20no%2013675910198240%7C%7Citem1367591023738
Verb(s)	GET
Input Object	
Output Object	DataItem

XML Response

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <v2:DataItem xmlns:v2="http://www.axeda.com/services/v2"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="model
4  no 13675910198240||item1367591023738" systemId="39"
5  label="item1367591023738" detail="STRING"
6  restUrl="http://jtelarico-
7  dt7.axeda.com:8080/services/v2/rest/dataItem/39">
8    <v2:name>item1367591023738</v2:name>
9    <v2:alias>alias1367591023738</v2:alias>
10   <v2:description>item description</v2:description>
11   <v2:model id="model no 13675910198240" systemId="79"
12 label="standalone" detail="model no 13675910198240"
13 restUrl="http://jtelarico-
14 dt7.axeda.com:8080/services/v2/rest/model/79"/>
15   <v2:type>STRING</v2:type>
```

```
16      <v2:readOnly>false</v2:readOnly>
        <v2:visible>true</v2:visible>
        <v2:forwarded>false</v2:forwarded>
        <v2:storage>STORED</v2:storage>
        <v2:stringMinLength>1</v2:stringMinLength>
        <v2:stringMaxLength>100</v2:stringMaxLength>
        <v2:analogRangeLowerLimit>1.0</v2:analogRangeLowerLimit>

        <v2:analogRangeUpperLimit>3.0</v2:analogRangeUpperLimit>

    </v2:DataItem>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/DataItemBridge.html>

DeploymentBridge

Deployments control which Software Packages are delivered to one or more Assets.

This section provides an overview of creating and manipulating Deployment objects through the Axeda API using the DeploymentBridge.

Deployment objects can contain properties that reference the Software Package to deploy, the time of deployment, the type and priority of the deployment, expiration date, status, dependencies, etc. For a full description of Deployment methods and properties, see the Axeda API Specification.

Deployments can either be manual or automatic. The main differentiator is that manual Deployments are tied to a fixed set of Assets, whereas automatic Deployments will allow Assets – as they become available – to receive the specified package.

As with Software Packages, Deployments can have expiration dates, and can also have dependencies. The only dependency type available to Deployments is DefinedConfiguration, which is formatted in the same manner as Software Packages.

The DeploymentBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for DeploymentBridge objects. You can also use the DeploymentBridge to redeploy Deployments.

Methods

The following table provides a brief overview of the available DeploymentBridge methods. For a full description of DeploymentBridge methods, see the Axeda API Specification.

Method	Description
cancel	Cancels the specified Deployment
create	Creates a new Deployment.
delete	Deletes a Deployment.
find (by criteria)	Finds Deployments based on search criteria.
find (by IDs)	Finds Deployments based on a list of identifiers.
find (by alternate ID)	Finds a Deployment based on the alternate identifier.
findById	Finds a Deployment based on its platform identifier.
findOne	Returns the first Deployment found that meets specified search criteria.
generateAlternateId	Returns a unique alternate identifier for the specified Deployment object.
redeploy (by list)	Redeploys the given Deployment for the referenced assetList in order to force the package mapped to the Deployment to be delivered and executed again -- assuming the

	deployment is still active and not expired.
redeploy (all, with forceRedelivery option)	Redeploys the given Deployment for all Assets the Deployment is mapped to, assuming the deployment is still eligible to be deployed. If the boolean parameter forceRedelivery is set to true, all Assets which are eligible to receive the packages will have them redelivered and executed. If forceRedelivery is set to false, only Assets that have not received the package in the previous attempt will have the deployment re-attempted.
toString	Generates a string representing a specified Deployment object.
update	Updates an existing Deployment.

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/DeploymentBridge.html>

ExpressionRuleBridge

You can use ExpressionRules to define your own business rules based on If-Then-Else constructs. An ExpressionRule is executed when its trigger occurs. Available triggers include alarm messages, system timer messages, or a change in the value of a DataItem.

When an ExpressionRule is triggered, its If expression is evaluated to a boolean value (true or false, or 1 or 0). The If expression may be composed of variables, operators, and functions. The Then expression will be executed when the If condition is evaluated to true, otherwise the Else expression will be executed. Both the Then and Else expressions should point to an action to be executed.

Expression Rules can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating Expression Rules through the API using the ExpressionRuleBridge object, and also lists the available REST calls.

The ExpressionRuleBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for ExpressionRule objects. You can also use the ExpressionRuleBridge to validate expressions.

Methods

The following table provides a brief overview of the ExpressionRuleBridge methods, along with available REST calls. For a full description of ExpressionRuleBridge methods, see the Axeda API Specification.

Method	Description	REST Call
create	Creates a new Expression Rule.	PUT <server>/services/v2/rest/expressionR **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/expressionR
delete	Deletes an Expression Rule.	DELETE <server>/services/v2/rest/expressionRule/id/{id}
find (by IDs)	Finds models based on a list of identifiers.	POST <server>/services/v2/rest/expressionRule/findByIds
find (by criteria)	Finds Expression Rules based on search criteria.	POST <server>/services/v2/rest/expressionRule/find
find (by alternate ID)	Finds an Expression Rule based on the alternate identifier.	GET <server>/services/v2/rest/expressionRule/id/{id}
findById	Finds an Expression Rule based on its platform	GET <server>/services/v2/rest/expressionR

		ule/id/{id}
findOne	Returns the first Expression Rule found that meets specified search criteria.	POST <server>/services/v2/rest/expressionRule/findOne
generateAlternateId	Returns a unique alternate identifier for the specified ExpressionRule object.	N/A
toString	Generates a string representing a specified Expression Rule.	N/A
update	Updates an existing Expression Rule.	POST <server>/services/v2/rest/expressionRule/id/{id}
validateExpressionRule	Validates the expressions in the specified Expression Rule.	POST <server>/services/v2/rest/expressionRule/validate

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using Expression Rule Collections.

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/expressionRule/bulk/create
bulkDelete	POST <server>/services/v2/rest/expressionRule/bulk/delete
bulkSave	POST <server>/services/v2/rest/expressionRule/bulk/save
bulkUpdate	POST <server>/services/v2/rest/expressionRule/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExpressionRuleBridge.html>

Create an ExpressionRule

Endpoint	/services/v2/rest/expressionRule
Verb(s)	PUT
Input Object	ExpressionRule
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExpressionRule xmlns="http://www.axeda.com/services/v2">

  <name>name1367257733170</name>
  <type>DATA</type>
  <description>description1367257733170</description>
  <ifExpression>true</ifExpression>
  <thenExpression>abs(1)</thenExpression>
  <elseExpression>Now()</elseExpression>
  <consecutive>true</consecutive>
  <applyToAll>false</applyToAll>
  <enabled>false</enabled>
  <standalone>true</standalone>
  <associatedModels>
    <model systemId="24"/>
  </associatedModels>
  <includedAssets>
    <asset systemId="35"/>
    <asset systemId="36"/>
  </includedAssets>
  <excludedAssets>
    <asset systemId="37"/>
  </excludedAssets>
</ExpressionRule>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="false">
  <v2:succeeded/>
  <v2:failures>
    <v2:failure code="ILLEGAL_ARGUMENT">
      <v2:ref>name1367257733170</v2:ref>
      <v2:message>The argument is invalid. Asset with id 36
does not exist in the system</v2:message>
      <v2:sourceOfFailure>unspecified</v2:sourceOfFailure>

    </v2:failure>
  </v2:failures>
</v2:ExecutionResult>
```

Update an ExpressionRule

Endpoint	/services/v2/rest/expressionRule/id/87
Verb(s)	POST
Input Object	
Output Object	

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExpressionRule xmlns="http://www.axeda.com/services/v2"
systemId="87">
  <name>name1367257734609</name>
  <type>DATA</type>
  <description>description1367257734609</description>
  <ifExpression>true</ifExpression>
  <thenExpression>abs(1)</thenExpression>
  <elseExpression>Now()</elseExpression>
  <consecutive>true</consecutive>
  <applyToAll>false</applyToAll>
  <enabled>false</enabled>
  <standalone>true</standalone>
  <associatedModels>
    <model systemId="24"/>
  </associatedModels>
  <includedAssets>
    <asset systemId="35"/>
    <asset systemId="36"/>
  </includedAssets>
  <excludedAssets>
    <asset systemId="37"/>
  </excludedAssets>
</ExpressionRule>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="false">
  <v2:succeeded/>
  <v2:failures>
    <v2:failure code="ILLEGAL_ARGUMENT">
      <v2:ref>87</v2:ref>
      <v2:message>The argument is invalid. Asset with id 36
does not exist in the system</v2:message>
      <v2:sourceOfFailure>unspecified</v2:sourceOfFailure>

    </v2:failure>
  </v2:failures>
</v2:ExecutionResult>
```

Delete an ExpressionRule

Endpoint	/services/v2/rest/expressionRule/id/90
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>90</v2:ref>
      <v2:id>90</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find an ExpressionRule by Criteria

Endpoint	/services/v2/rest/expressionRule/find
Verb(s)	POST
Input Object	ExpressionRuleCriteria
Output Object	FindExpressionRuleResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExpressionRuleCriteria
xmlns="http://www.axeda.com/services/v2" pageSize="10"
pageNumber="1">
  <description>1367591077167common description</description>

</ExpressionRuleCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindExpressionRuleResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="2">
  <v2:criteria pageSize="10" pageNumber="1">
    <v2:description>1367591077167common
description</v2:description>
  </v2:criteria>
  <v2:expressionRules>
    <v2:expressionRule xsi:type="v2:ExpressionRule"
id="name1367591077167a" systemId="45"
label="name1367591077167a" detail="Data"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/expressionRule/45">
      <v2:name>name1367591077167a</v2:name>
      <v2:type>DATA</v2:type>
      <v2:description>1367591077167common
description</v2:description>
      <v2:ifExpression>true</v2:ifExpression>
      <v2:thenExpression>abs(1)</v2:thenExpression>
      <v2:elseExpression>Now()</v2:elseExpression>
      <v2:consecutive>true</v2:consecutive>
      <v2:applyToAll>false</v2:applyToAll>
      <v2:enabled>false</v2:enabled>
      <v2:createdBy>admin</v2:createdBy>
      <v2:lastModifiedBy>admin</v2:lastModifiedBy>
      <v2:creationDate>2013-05-03T10:24:37.939-04:00</v2:crea
```

```
tionDate>
      <v2:lastModificationDate>2013-05-03T10:24:37.939-04:00<
/v2:lastModificationDate>
      <v2:standalone>true</v2:standalone>
      <v2:associatedModels>
        <v2:model id="model_nr_NCAII13675910609941"
systemId="83" label="standalone"
detail="model_nr_NCAII13675910609941"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/83"/>
      </v2:associatedModels>
      <v2:includedAssets>
        <v2:asset id="model_nr_NCAII13675910609940||
13675910610443" systemId="449"
label="model_nr_NCAII13675910609940" detail="13675910610443"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/449"/>
      </v2:includedAssets>
      <v2:excludedAssets>
        <v2:asset id="model_nr_NCAII13675910609940||
13675910610444" systemId="450"
label="model_nr_NCAII13675910609940" detail="13675910610444"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/450"/>
        <v2:asset id="model_nr_NCAII13675910609941||
13675910610440" systemId="451"
label="model_nr_NCAII13675910609941" detail="13675910610440"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/451"/>
      </v2:excludedAssets>
    </v2:expressionRule>
    <v2:expressionRule xsi:type="v2:ExpressionRule"
id="name1367591077167b" systemId="46"
label="name1367591077167b" detail="Data"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/expressionRule/46">
      <v2:name>name1367591077167b</v2:name>
      <v2:type>DATA</v2:type>
      <v2:description>1367591077167common
description</v2:description>
      <v2:ifExpression>true</v2:ifExpression>
      <v2:thenExpression>abs(1)</v2:thenExpression>
      <v2:elseExpression>Now()</v2:elseExpression>
      <v2:consecutive>true</v2:consecutive>
      <v2:applyToAll>false</v2:applyToAll>
      <v2:enabled>false</v2:enabled>
      <v2:createdBy>admin</v2:createdBy>
      <v2:lastModifiedBy>admin</v2:lastModifiedBy>
      <v2:creationDate>2013-05-03T10:24:37.979-04:00</v2:crea
tionDate>
      <v2:lastModificationDate>2013-05-03T10:24:37.979-04:00<
/v2:lastModificationDate>
      <v2:standalone>true</v2:standalone>
      <v2:associatedModels>
        <v2:model id="model_nr_NCAII13675910609941"
systemId="83" label="standalone"
detail="model_nr_NCAII13675910609941"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/83"/>
      </v2:associatedModels>
      <v2:includedAssets>
```

```
        <v2:asset id="model_nr_NCAII13675910609940||
13675910610443" systemId="449"
label="model_nr_NCAII13675910609940" detail="13675910610443"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/449"/>
      </v2:includedAssets>
      <v2:excludedAssets>
        <v2:asset id="model_nr_NCAII13675910609940||
13675910610444" systemId="450"
label="model_nr_NCAII13675910609940" detail="13675910610444"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/450"/>
        <v2:asset id="model_nr_NCAII13675910609941||
13675910610440" systemId="451"
label="model_nr_NCAII13675910609941" detail="13675910610440"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/asset/451"/>
      </v2:excludedAssets>
    </v2:expressionRule>
  </v2:expressionRules>
</v2:FindExpressionRuleResult>
```

Validate an ExpressionRule

Endpoint	/services/v2/rest/expressionRule/validate
Verb(s)	POST
Input Object	ExpressionRule
Output Object	ExpressionRuleValidationResult

XML Request

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ExpressionRule xmlns="http://www.axeda.com/services/v2"
3 systemId="1">
4   <ifExpression>ceil(1)</ifExpression>
  </ExpressionRule>
```

XML Response

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <v2:ExpressionRuleValidationResult xmlns:v2
3 ="http://www.axeda.com/services/v2"
4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5   <v2:ref>1</v2:ref>
6   <v2:successful>true</v2:successful>
    <v2:ifResult>OK</v2:ifResult>
  </v2:ExpressionRuleValidationResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExpressionRuleBridge.html>

ExtendedListBridge

An Extended List represents an ordered collection of string entries. The ExtendedList object is a simple, named, stand-alone list of Strings. You can search for an ExtendedList object by name, or by content by using the entry property in ExtendedListCriteria. An ExtendedList object can contain up to 1,000 entries per list and up to 256 characters per entry. Only unique values will be stored in the list, which means an ExtendedList behaves like a Set. Any duplicate values supplied during Create, Update or Append operations are discarded.

An Extended List lets you add new virtual columns and virtual tables to the Axeda Platform, so you can define new properties that are specific to your Assets, and then track and manage the associated values (metadata).

An Extended List is always of type: *java.util.ArrayList<String>.* Note that an application can append and delete entries from a single list at one time.

Extended Lists can be created through the Axeda API, or via a REST call over HTTP.

Methods

The following table provides a brief overview of the available ExtendedListBridge methods, along with available REST calls. For a full description of ExtendedListBridge methods, see the Axeda API Specification.

Method	Description	REST Call
append	Appends (adds) a value to an existing Extended List.	PUT <server>/services/v2/rest/extendedList/id/{id}
create	Creates a new Extended List. **Note:** When an ExtendedList is created, the Name value is also used as the systemId. Therefore, if another ExtendedList is created using the same name, the new ExtendedList will overwrite the existing ExtendedList.	PUT <server>/services/v2/rest/extendedList **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/extendedList
delete	Deletes an Extended List. **Note:** When the delete method is invoked, the contents of an ExtendedList are deleted, but not the ExtendedList itself. A search for a deleted ExtendedList will return an empty list.	DELETE <server>/services/v2/rest/extendedList/id/{id}
find (by criteria)	Finds ExtendendList objects based on search criteria.	POST <server>/services/v2/rest/extendedList/find

find (by IDs)	Finds ExtendendList objects based on a list of IDs.	N/A
findById	Finds an ExtendedList object based on its platform identifier (systemId). **Note:** for ExtendedList objects, name, systemId, and alternateId are all the same value.	GET <server>/services/v2/rest/extendedList/id/{id}
find (by alternate ID)	Finds an ExtendedList object based on the alternate identifier. **Note:** for ExtendedList objects, name, systemId, and alternateId are all the same value.	GET <server>/services/v2/rest/extendedList/id/{id}
findOne	Returns the first ExtendedList object found that meets specified search criteria.	POST <server>/services/v2/rest/extendedList/findOne
generateAlternateId	Returns a unique alternate identifier for the specified Extended List.	N/A
remove	Removes a specified value from an Extended List.	DELETE <server>/services/v2/rest/extendedList/id/{id}
toString	Generates a string representing a specified Extended List.	N/A
update	Updates an existing Extended List.	POST <server>/services/v2/rest/extendedList/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using ExtendedList Collections.

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/extendedList/bulk/create
bulkDelete	POST <server>/services/v2/rest/extendedList/bulk/delete
bulkSave	POST <server>/services/v2/rest/extendedList/bulk/save
bulkUpdate	POST <server>/services/v2/rest/extendedList/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExtendedListBridge.html>

Create an ExtendedList

```
1  package sdkexamples.extendedlist
2
3  import static com.axeda.sdk.v2.dsl.Bridges.*
4  import com.axeda.services.v2.*
5
6  def extList1 = new ExtendedList(
7      name: "Roles list",
8      list: ["System Administrator", "Database Administrator", "Technician"]
9  )
10
11 def result = extendedListBridge.create(extList1)
12 if (result.successful) {
13   println "The extended list was created"
14 }
15 else {
16   assert result.failures.any() { it.code == "NOT_UNIQUE" }
17 }
```

Delete an ExtendedList

```
1  package sdkexamples.extendedlist
2
3  import static com.axeda.sdk.v2.dsl.Bridges.*
4  import com.axeda.services.v2.*
5
6  def extList1 = new ExtendedList(id: "Countries list")
7
8  def result = extendedListBridge.delete(extList1)
9  if (result.successful) {
10   println "The extended list was deleted"
11 }
12 else {
13   assert result.failures.every { it.code == "OBJECT_NOT_FOUND" }
14 }
```

Find ExtendedList by Alternate ID

```
1  package sdkexamples.extendedlist
2
3  import static com.axeda.sdk.v2.dsl.Bridges.*
4
5  def foundExtendedList = extendedListBridge.find("RatePlans list")
6  assert foundExtendedList
```

Find an ExtendedList by System ID

```
1  package sdkexamples.extendedlist
2
3  import com.axeda.drm.sdk.customobject.*
```

```
import static com.axeda.sdk.v2.dsl.Bridges.*

String extListId = Call.parameters.extendedListId

def foundExtendedList = extendedListBridge.findById(extListId)
assert foundExtendedList
```

Find One ExtendedList

```
package sdkexamples.extendedlist

import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def findResult = extendedListBridge.findOne(
    new ExtendedListCriteria(name : "Colors list"))
assert findResult
```

Save an ExtendedList

```
package sdkexamples.extendedlist

import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def extList1 = new ExtendedList(
    id: "RatePlans list",
    name: "RatePlans list",
    list: ["RatePlan1", "RatePlan2"]
)
def result = extendedListBridge.save(extList1)
assert result.successful
```

Update an ExtendedList

```
package sdkexamples.extendedlist

import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def extList1 = new ExtendedList(
    id: "RatePlans list",
    name: "RatePlans list",
    list: ["RatePlan1", "RatePlan2"]
)

def result = extendedListBridge.update(extList1)
assert result.successful
```

Bulk Create

```
package sdkexamples.extendedlist

import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def extList1 = new ExtendedList(
```

```
    name: "Profiles list",
    list: ["profile1", "profile2", "profile3"]
)

def extList2 = new ExtendedList(
    name: "Statuses list",
    list: ["active", "inactive", "standby"]
)

def result = extendedListBridge.create([extList1, extList2])
if (result.successful) {
  println "The extended lists were created"
}
else {
  assert result.failures.every { it.code == "NOT_UNIQUE" }
}
```

Bulk Delete

```
package sdkexamples.extendedlist

import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def extList1 = new ExtendedList(id: "External systems list")
def extList2 = new ExtendedList(id: "Items whitelist")

def result = extendedListBridge.delete([extList1, extList2])
if (result.successful) {
  println "The extended lists were deleted"
}
else {
  assert result.failures.every { it.code == "OBJECT_NOT_FOUND" }
}
```

Bulk Save

```
package sdkexamples.extendedlist

import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def extList1 = new ExtendedList(
    id: "RatePlans list",
    name: "RatePlans list",
    list: ["RatePlan1", "RatePlan2", "RatePlan3", "RatePlan4"]
)

def extList2 = new ExtendedList(
    name: "Cities list",
    list: ["New York", "Tokyo", "London"]
)

def result = extendedListBridge.save([extList1, extList2])
if (result.successful) {
  println "One extended list was created and one was updated"
}
else {
  assert result.failures.every() { it.code == "NOT_UNIQUE" }
```

```
}
```

Bulk Update

```
package sdkexamples.extendedlist

import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def extList1 = new ExtendedList(
    id: "RatePlans list",
    name: "RatePlans list",
    list: ["RatePlan1", "RatePlan2", "RatePlan3", "RatePlan4"]
)

def extList2 = new ExtendedList(
    id: "Colors list",
    name: "Colors list",
    list: ["red", "orange", "black"]
)

def result = extendedListBridge.update([extList1, extList2])
assert result.successful
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExtendedListBridge.html>

Create an ExtendedList

Endpoint	/services/v2/rest/extendedList
Verb(s)	PUT
Input Object	ExtendedList
Output Object	ExecutionResult

XML Request

```
REQUEST         :
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedList xmlns="http://www.axeda.com/services/v2">
  <name>lst_0011367258543479</name>
  <list>
    <item>item0</item>
    <item>item1</item>
    <item>item2</item>
    <item>item3</item>
    <item>item4</item>
    <item>item5</item>
    <item>item6</item>
    <item>item7</item>
    <item>item8</item>
    <item>item9</item>
    <item>item10</item>

//lines 11-990 omitted in this sample for brevity

    <item>item991</item>
    <item>item992</item>
    <item>item993</item>
    <item>item994</item>
    <item>item995</item>
    <item>item996</item>
    <item>item997</item>
    <item>item998</item>
    <item>item999</item>
  </list>
</ExtendedList>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>lst_0011367258543479</v2:ref>
      <v2:id>lst_0011367258543479</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Update an ExtendedList

Endpoint	/services/v2/rest/extendedList/id/lst_0031367258543479
Verb(s)	POST
Input Object	ExtendedList
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedList xmlns="http://www.axeda.com/services/v2" id="lst_
0031367258543479" systemId="lst_0031367258543479"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedList/lst_
0031367258543479">
  <name>lst_0031367258543479</name>
  <list>
    <item>a</item>
    <item>b</item>
    <item>c</item>
    <item>d</item>
    <item>e</item>
  </list>
</ExtendedList>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>lst_0031367258543479</v2:ref>
      <v2:id>lst_0031367258543479</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Save an ExtendedList

Endpoint	/services/v2/rest/extendedList
Verb(s)	POST
Input Object	DataItem
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedList xmlns="http://www.axeda.com/services/v2">
  <name>lst_0051367258543479</name>
  <list>
    <item>a</item>
    <item>b</item>
    <item>c</item>
```

```
  </list>
</ExtendedList>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>lst_0051367258543479</v2:ref>
      <v2:id>lst_0051367258543479</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete an ExtendedList

Endpoint	/services/v2/rest/extendedList/id/lst_0061367258543479
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>lst_0061367258543479</v2:ref>
      <v2:id>lst_0061367258543479</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find One ExtendedList

Endpoint	/services/v2/rest/extendedList/findOne
Verb(s)	POST
Input Object	ExtendedListCriteria
Output Object	ExtendedList

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedListCriteria xmlns="http://www.axeda.com/services/v2">

  <name>lst_010_*1367591096536</name>
</ExtendedListCriteria>
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

Response

```
<v2:ExtendedList xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="lst_
010_11367591096536" systemId="lst_010_11367591096536"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedList/lst_010_
11367591096536">

  <v2:name>lst_010_11367591096536</v2:name>
  <v2:list>
    <v2:item>a</v2:item>
    <v2:item>b</v2:item>
    <v2:item>c</v2:item>
  </v2:list>
</v2:ExtendedList>
```

Find an ExtendedList by Alternate ID

Endpoint	Find an ExtendedList by Alternate ID
Verb(s)	GET
Input Object	
Output Object	ExtendedList

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExtendedList xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="lst_
0081367591096536" systemId="lst_0081367591096536"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedList/lst_
0081367591096536">

  <v2:name>lst_0081367591096536</v2:name>
  <v2:list>
    <v2:item>a</v2:item>
    <v2:item>b</v2:item>
    <v2:item>c</v2:item>
  </v2:list>
</v2:ExtendedList>
```

Append to an ExtendedList

Endpoint	/services/v2/rest/extendedList/lst_0171367591096536
Verb(s)	PUT
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>lst_0171367591096536</v2:ref>
```

```
      <v2:id>lst_0171367591096536</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Remove from an ExtendedList

Endpoint	/services/v2/rest/extendedList/lst_0181367591096536/b
Verb(s)	DELETE
Input Object	
Output Object	

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExtendedListBridge.html>

ExtendedMapBridge

An Extended Map represents an ordered collection of String name-value pairs. Any duplicate names are discarded. The Extended Map is a collection of names (String keys) and values with up to 1,000 name-value pairs per list. Only unique values will be stored in the map. If duplicate values are supplied, only the last supplied value will be stored. Also, if a duplicate name (key) is used, the value will overwrite the existing value.

Any duplicate key will result in the value of the existing key being overwritten.

The name of the map can be up to 64 characters long. Each name in a name-value pair can be up to 128 characters long, and each value can be up to 256 characters long.

You can use Extended Maps as extensibility points for application development — for example, to store frequently accessed maps of information that are accessed within an application. When combined with the ExtendedListBridge object, robust graphs of data can be stored.

A named map is always of type: *java.util.HashMap<String,String>*. An application can append and delete entries on a single given map at one time.

Extended Maps can be created through the Axeda API, or via a REST call over HTTP.

Methods

The following table provides a brief overview of the available ExtendedMapBridge methods, along with available REST calls. For a full description of ExtendedMapBridge methods, see the Axeda API Specification.

Method	Description	REST Call
append	Appends (adds) a value to an existing Extended Map.	PUT <server>/services/v2/rest/extendedMap/id/{id}
create	Creates a new Extended Map. **Note:** When an ExtendedMap is created, the Name value is also used as the systemId. Therefore, if another ExtendedMap is created using the same name, the new ExtendedMap will overwrite the existing ExtendedMap.	PUT <server>/services/v2/rest/extendedMap **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/extendedMap
delete	Deletes an Extended Map.	DELETE

	Note: When the delete method is invoked, the contents of an ExtendedMap are deleted, but not the ExtendedMap itself. A search for a deleted ExtendedMap will return an empty map.	<server>/services/v2/rest/extendedMap/id/{id}
find (by criteria)	Finds ExtendendMap objects based on search criteria.	POST <server>/services/v2/rest/extendedMap/find
find (by IDs)	Finds ExtendendMap objects based on a list of IDs.	N/A
findAsMap	Returns the contents of an ExtendedMap as a java.util.Map. This allows a symmetrical relationship between the find and create where Maps are used as the primary construct.	
findById	Finds an ExtendedMap object based on its platform identifier. **Note:** for ExtendedMap objects, name, systemId, and alternateId are all the same value.	GET <server>/services/v2/rest/extendedMap/id/{id}
find (by alternate ID)	Finds an ExtendedMap object based on the alternate identifier. **Note:** for ExtendedMap objects, name, systemId, and alternateId are all the same value.	GET <server>/services/v2/rest/extendedMap/id/{id}
findOne	Returns the first ExtendedMap object found that meets specified search criteria.	POST <server>/services/v2/rest/extendedMap/findOne
fromMap	Constructs an Extended Map instance out of a map.	
getValue	Returns the value for the specified name (key) in the referenced Extended map.	
generateAlternateId	Returns a unique alternate identifier for the specified Extended Map.	N/A
remove	Removes a specified value from an Extended Map.	DELETE <server>/services/v2/rest/extendedList/id/{id}
toString	Generates a string representing a specified Extended Map.	N/A
update	Updates an existing Extended Map.	POST <server>/services/v2/rest/extendedMap/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using ExtendedMap Collections.

Operation	REST Call

bulkCreate	POST <server>/services/v2/rest/extendedMap/bulk/create
bulkDelete	POST <server>/services/v2/rest/extendedMap/bulk/delete
bulkSave	POST <server>/services/v2/rest/extendedMap/bulk/save
bulkUpdate	POST <server>/services/v2/rest/extendedMap/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExtendedMapBridge.html>

Create an ExtendedMap

```
import static com.axeda.sdk.v2.dsl.Bridges.*

/*  create the ExtendedMap using the fromMap method */
def dollarExchangeMap = extendedMapBridge.fromMap("Dollar exchange rate
map",
                                                  ["USD": "1", "RON":
"3.3", "GBP": "0.66", "NOK": "5.76"])
def result = extendedMapBridge.create(dollarExchangeMap)

if (result.isSuccessful()) {
  println "The dollar exchange rates map was created "
}
else {
  assert result.failures.every {it.code == 'NOT_UNIQUE'}
}
```

Delete an ExtendedMap

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def deleteResult = extendedMapBridge.delete(new ExtendedMap(id: "Example
assert deleteResult.succeeded
```

Find as Map

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def foundMap = extendedMapBridge.findAsMap("Example State names")
assert foundMap
println "The state that has the abbreviation 'MA' is : ${foundMap['MA']}"
```

Find an ExtendedMap by Alternate ID

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def foundMap = extendedMapBridge.findById("Example ISO country codes")
assert foundMap
println "The country code for France is : ${foundMap.map.find({it.name ==
'France'})?.value}"
```

Find an ExtendedMap by Criteria

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.ExtendedMapCriteria

def foundResult = extendedMapBridge.find(new ExtendedMapCriteria(key:
"United States of America"))
```

```
assert foundResult.maps.any {it.name == 'Example ISO country codes'}
```

Find an ExtendedMap by System ID

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def foundMap = extendedMapBridge.findById("Example ISO country codes")
assert foundMap
println "The country code for Romania is : ${foundMap.map.find({it.name == 'Romania'})?.value}"
```

Find One ExtendedMap

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.ExtendedMapCriteria

def foundMap = extendedMapBridge.findOne(new ExtendedMapCriteria(name:
assert foundMap
```

Create from Map

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def dollarExchangeMap = extendedMapBridge.fromMap(
    "Dollar exchange rate map",
    ["USD": "1", "RON": "3.3", "GBP": "0.66", "NOK": "5.76"])
```

Get Value

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def foundValue = extendedMapBridge.getValue("Example State names", "MA")
assert foundValue
println "The state that has the abbreviation 'MA' is : ${foundValue}"
```

Remove Value

```
import static com.axeda.sdk.v2.dsl.Bridges.*

/* we don't support Texas any more.  */
extendedMapBridge.remove("Example State names", "TX")
```

Save an ExtendedMap

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def isoCountriesMap = extendedMapBridge.fromMap(
    "Example ISO country codes",
    ["United States of America": "US", "Romania": "RO", "France": "FR",
isoCountriesMap.id = "Example ISO country codes"
def updateResult = extendedMapBridge.save(isoCountriesMap)

assert updateResult.successful
```

Update an ExtendedMap

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def isoCountriesMap = extendedMapBridge.fromMap(
    "Example ISO country codes",
    ["United States of America": "US", "Romania": "RO", "France": "FR",
"Italy": "IT", "Spain": "ES"])
isoCountriesMap.id = "Example ISO country codes"

def updateResult = extendedMapBridge.update(isoCountriesMap)

assert updateResult.successful
```

Append an ExtendedMap

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def result = extendedMapBridge.append("Example State names", "MI",
assert result.successful
```

Bulk Create

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def roleRightsMap = extendedMapBridge.fromMap(
    "Example role rights map", ["Admin": "CRUD", "View": "R", "Super-
def userRoleMap = extendedMapBridge.fromMap(
    "Example users to role", ["admin": "Admin", "guest": "View", "me":
def bulkCreateResult = extendedMapBridge.create([roleRightsMap,

if (bulkCreateResult.successful) {
  println "The extended maps were created"
}
else {
  assert bulkCreateResult.failures.every {it.code == 'NOT_UNIQUE'}
}
```

Bulk Delete

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def deleteResult = extendedMapBridge.delete([
    new ExtendedMap(id: "Example Settings map"),
    new ExtendedMap(id: "Example Ip ranges map")])

assert deleteResult.successful
```

Bulk Save

```
import static com.axeda.sdk.v2.dsl.Bridges.*

/*  this map will be created*/
def preferredCategories = extendedMapBridge.fromMap(
```

```
    "Example preferred categories",
    ["Jhon Doe": "music,shirts", "Mike Doe": "computers,books"])

/*  this map will be updated  */
def statesNames = extendedMapBridge.fromMap(
    "Example State names",
    ["MA" : "Massachusetts", "NY" : "New York", "AK" : "Alaska", "TX" :
)
statesNames.id ="Example State names"

def saveResult = extendedMapBridge.save([preferredCategories,
assert saveResult.successful
```

Bulk Update

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def isoCountriesMap = extendedMapBridge.fromMap(
    "Example ISO country codes",
    ["United States of America": "US", "Romania" : "RO", "France": "FR",
"Italy": "IT", "Ireland": "IE"])
isoCountriesMap.id = "Example ISO country codes"

def statesNames = extendedMapBridge.fromMap(
    "Example State names",
    ["MA" : "Massachusetts", "NY" : "New York", "AK" : "Alaska", "TX" :
)
statesNames.id ="Example State names"

def updateResult = extendedMapBridge.update([isoCountriesMap,

assert updateResult.successful
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExtendedMapBridge.html>

Create an ExtendedMap

Endpoint	/services/v2/rest/extendedMap
Verb(s)	PUT
Input Object	ExtendedMap
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedMap xmlns="http://www.axeda.com/services/v2">
  <name>map_0011367591142304</name>
  <map>
    <namedValue>
      <name>1367591142304_map_001_key_0</name>
      <value>1367591142304_map_001_value_0</value>
    </namedValue>
    <namedValue>
      <name>1367591142304_map_001_key_1</name>
      <value>1367591142304_map_001_value_1</value>
    </namedValue>
    <namedValue>
      <name>1367591142304_map_001_key_2</name>
      <value>1367591142304_map_001_value_2</value>
    </namedValue>
  </map>
</ExtendedMap>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded>
    <v2:success>
      <v2:ref>map_0011367591142304</v2:ref>
      <v2:id>map_0011367591142304</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Update an ExtendedMap

Endpoint	/services/v2/rest/extendedMap/id/map_0021367591142304
Verb(s)	POST
Input Object	ExtendedMap
Output	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedMap xmlns="http://www.axeda.com/services/v2" id="map_
0021367591142304" systemId="map_0021367591142304"
restUrl="http://jtelarico-
```

```
dt7.axeda.com:8080/services/v2/rest/extendedMap/map_
  <name>map_0021367591142304</name>
  <map>
    <namedValue>
      <name>1367591142304_map_002_key_0</name>
      <value>1367591142304_map_002_value_0</value>
    </namedValue>
    <namedValue>
      <name>1367591142304_map_002_key_1</name>
      <value>1367591142304_map_002_value_1</value>
    </namedValue>
    <namedValue>
      <name>ChangedName</name>
      <value>ChangedValue</value>
    </namedValue>
    <namedValue>
      <name>Extra1</name>
      <value>Value1</value>
    </namedValue>
    <namedValue>
      <name>Extra2</name>
      <value>Value2</value>
    </namedValue>
  </map>
</ExtendedMap>
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>map_0021367591142304</v2:ref>
      <v2:id>map_0021367591142304</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete an ExtendedMap

Endpoint	/services/v2/rest/extendedMap/id/map_0051367591142304
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded>
    <v2:success>
      <v2:ref>map_0051367591142304</v2:ref>
      <v2:id>map_0051367591142304</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find an ExtendedMap by Criteria	
Endpoint	/services/v2/rest/extendedMap/find
Verb(s)	POST
Input Object	ExtendedMapCriteria
Output Object	FindExtendedMapResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedMapCriteria xmlns="http://www.axeda.com/services/v2">
  <name>1367591142304_map_find*</name>
</ExtendedMapCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindExtendedMapResult xmlns:v2
="http://www.axeda.com/services/v2"
  <v2:criteria>
    <v2:name>1367591142304_map_find*</v2:name>
  </v2:criteria>
  <v2:maps>
    <v2:map xsi:type="v2:ExtendedMap" id="1367591142304
_map_finda11367591142304" systemId="1367591142304
_map_finda11367591142304" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedMap/1367591142304
_map_finda11367591142304">
      <v2:name>1367591142304_map_finda11367591142304</v2:name>
      <v2:map>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_finda1_key_
          <v2:value>1367591142304_1367591142304_map_finda1
_value_0</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_finda1_key_
          <v2:value>1367591142304_1367591142304_map_finda1
_value_1</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_finda1_key_
          <v2:value>1367591142304_1367591142304_map_finda1
_value_2</v2:value>
        </v2:namedValue>
      </v2:map>
    </v2:map>
    <v2:map xsi:type="v2:ExtendedMap" id="1367591142304
_map_finda21367591142304" systemId="1367591142304
_map_finda21367591142304" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedMap/1367591142304
_map_finda21367591142304">
      <v2:name>1367591142304_map_finda21367591142304</v2:name>
      <v2:map>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_finda2_key_
          <v2:value>1367591142304_1367591142304_map_finda2
_value_0</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_finda2_key_
```

```
          <v2:value>1367591142304_1367591142304_map_finda2
_value_1</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_finda2_key_
          <v2:value>1367591142304_1367591142304_map_finda2
_value_2</v2:value>
        </v2:namedValue>
      </v2:map>
    </v2:map>
    <v2:map xsi:type="v2:ExtendedMap" id="1367591142304
_map_findb31367591142304" systemId="1367591142304
_map_findb31367591142304" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedMap/1367591142304
_map_findb31367591142304">
      <v2:name>1367591142304_map_findb31367591142304</v2:name>
      <v2:map>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_findb3_key_
          <v2:value>1367591142304_1367591142304_map_findb3
_value_0</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_findb3_key_
          <v2:value>1367591142304_1367591142304_map_findb3
_value_1</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_findb3_key_
          <v2:value>1367591142304_1367591142304_map_findb3
_value_2</v2:value>
        </v2:namedValue>
      </v2:map>
    </v2:map>
    <v2:map xsi:type="v2:ExtendedMap" id="1367591142304
_map_findb41367591142304" systemId="1367591142304
_map_findb41367591142304" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedMap/1367591142304
_map_findb41367591142304">
      <v2:name>1367591142304_map_findb41367591142304</v2:name>
      <v2:map>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_findb4_key_
          <v2:value>1367591142304_1367591142304_map_findb4
_value_0</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_findb4_key_
          <v2:value>1367591142304_1367591142304_map_findb4
_value_1</v2:value>
        </v2:namedValue>
        <v2:namedValue>
          <v2:name>1367591142304_1367591142304_map_findb4_key_
          <v2:value>1367591142304_1367591142304_map_findb4
_value_2</v2:value>
        </v2:namedValue>
      </v2:map>
    </v2:map>
  </v2:maps>
</v2:FindExtendedMapResult>
```

Find One Extended Map

Endpoint	/services/v2/rest/extendedMap/findOne
Verb(s)	POST
Input	ExtendedMapCriteria
Output Object	ExtendedMap

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExtendedMapCriteria xmlns="http://www.axeda.com/services/v2"
sortAscending="true" sortPropertyName="Name">
  <name>map_009*1367591142304</name>
</ExtendedMapCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExtendedMap xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="map_
009_11367591142304" systemId="map_009_11367591142304"
restUrl="http://jtelarico-
  <v2:name>map_009_11367591142304</v2:name>
  <v2:map>
    <v2:namedValue>
      <v2:name>1367591142304_map_009_1_key_0</v2:name>
      <v2:value>1367591142304_map_009_1_value_0</v2:value>
    </v2:namedValue>
    <v2:namedValue>
      <v2:name>1367591142304_map_009_1_key_1</v2:name>
      <v2:value>1367591142304_map_009_1_value_1</v2:value>
    </v2:namedValue>
    <v2:namedValue>
      <v2:name>1367591142304_map_009_1_key_2</v2:name>
      <v2:value>1367591142304_map_009_1_value_2</v2:value>
    </v2:namedValue>
  </v2:map>
</v2:ExtendedMap>
```

Find an ExtendedMap by System

Endpoint	/services/v2/rest/extendedMap/id/map_0071367591142304
Verb(s)	GET
Input Object	
Output Object	ExtendedMap

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExtendedMap xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="map_
0071367591142304" systemId="map_0071367591142304"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/extendedMap/map_
  <v2:name>map_0071367591142304</v2:name>
  <v2:map>
    <v2:namedValue>
      <v2:name>1367591142304_map_007_key_0</v2:name>
      <v2:value>1367591142304_map_007_value_0</v2:value>
```

```
    </v2:namedValue>
    <v2:namedValue>
      <v2:name>1367591142304_map_007_key_1</v2:name>
      <v2:value>1367591142304_map_007_value_1</v2:value>
    </v2:namedValue>
    <v2:namedValue>
      <v2:name>1367591142304_map_007_key_2</v2:name>
      <v2:value>1367591142304_map_007_value_2</v2:value>
    </v2:namedValue>
  </v2:map>
</v2:ExtendedMap>
```

Append to an ExtendedMap

Endpoint	/services/v2/rest/extendedMap/map_0391367591142304/newKey1
Verb(s)	PUT
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded>
    <v2:success>
      <v2:ref>map_0391367591142304</v2:ref>
      <v2:id>map_0391367591142304</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Back

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExtendedMapBridge.html>

ExtendedUIModuleBridge

An Extended UI Module allows you to create a custom module for the Asset Dashboard in the Axeda Console. You can write a Groovy script and add it as a Custom Object using the Axeda Configuration application. Alternatively, you can write an application and add it as a Custom Application using the Axeda Administration application. You can then configure the Custom Object or Custom Application as an Extended UI Module, using the Axeda Configuration application OR using the components of the Axeda V2 SDK described in this section.

The ExtendedUIModule object provides access to all of the features of the Extended UI Module feature in the Axeda Console. Using the methods of the ExtendedUIModuleBridge class or the operations of the ExtendedUIModule Web service, you can create, retrieve, update, and delete Extended UI Modules in the Axeda Platform. The ExtendedUIModule service is a RESTful Web service that provides access to all of the Extended UI Module-related behaviors. Each operation is the direct equivalent of a method of the ExtendedUIModuleBridge class.

Display of an Extended UI Module

You can specify the height of an Extended UI Module (from 50 to 1200 pixels) when using the Create, Update, or Save methods. The width of the module is determined by the width of the browser window and the content of the three columns on the Asset dashboard. For example, suppose the browser window is 1080 pixels wide and the left and right columns are each 230 pixels wide (they are always narrower than the center column). The center column (recommended for placement of Extended UI Modules) would be 620 pixels wide.

Once an Extended UI Module is added to the Axeda Platform, you need to use the Model Dashboard Layout and the Default Dashboard Layout pages of the Axeda Configuration application to add the module to the Asset dashboards (Axeda Service application). All Extended UI Modules have the suffix "*" to distinguish them from the default Axeda modules. A maximum of 10 Extended UI Modules are allowed.

Once the Extended UI Module is part of the layout, keep in mind that the module loads asynchronously so that it does not affect the total load time of the dashboard. While it is loading, the Extended UI Module appears as an empty frame with the height defined in the Extended UI Module and shows the message, "Loading…". Refer to the online help for the Axeda Configuration and Axeda Service applications for details.

To enable the use of Extended UI Modules in the Axeda Platform, a property that is set by default to false must be set to true in the configuration file for the Axeda Enterprise Server. Contact Axeda Technical Support for assistance. In addition, another property sets the loading timeout for Extended UI Modules. The default timeout is 30 seconds. If your module times out frequently (the message, "Failed to load," may appear), consider optimizing the loading of the module code or requesting a change to the timeout.

Internationalization

The Axeda Platform currently supports a user-configurable Locale setting. If an Extended UI Module has a type of Custom Object, then that user-configurable Locale must be passed to the Custom Object script to allow developers to generate locale-specific HTML, and so on. If an Extended UI Module has a type of Custom Application, then the locale must be passed to the target application to allow developers to create a locale-specific application.

Methods

The following table provides a brief overview of the available ExtendedUIModuleBridge methods, along with available REST calls. For a full description of ExtendedUIModuleBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
create	Creates a new ExtendedUIModule object.	PUT <server>/services/v2/rest/ExtendedUIModule **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/ExtendedUIModule
delete	Deletes a ExtendedUIModule object.	DELETE <server>/services/v2/rest/ExtendedUIModule/id/{id}
find (by criteria)	Finds ExtendedUIModule objects based on search	POST <server>/services/v2/rest/ExtendedUIModule/find
find (by IDs)	Finds ExtendedUIModule objects based on a list of	POST <server>/services/v2/rest/ExtendedUIModule/findByIds
find (by alternate ID)	Finds a ExtendedUIModule object based on the alternate identifier.	GET <server>/services/v2/rest/ExtendedUIModule/id/{id}
findById	Finds a ExtendedUIModule object based on its platform identifier.	GET <server>/services/v2/rest/ExtendedUIModule/id/{id}
findOne	Returns the first ExtendedUIModule object found that meets specified search criteria.	POST <server>/services/v2/rest/ExtendedUIModule/findOne
toString	Generates a string representing a specified ExtendedUIModule object.	N/A
update	Updates an existing ExtendedUIModule object.	POST <server>/services/v2/rest/ExtendedUIModule/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations:

Operation	REST Call

bulkCreate	POST <server>/services/v2/rest/ExtendedUIModule/bulk/create
bulkDelete	POST <server>/services/v2/rest/ExtendedUIModule/bulk/delete
bulkSave	POST <server>/services/v2/rest/ExtendedUIModule/bulk/save
bulkUpdate	POST <server>/services/v2/rest/ExtendedUIModule/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExtendedUIModuleBridge.html>

ExternalCredentialBridge

An External Credential is a collection of name-value pairs, some variable and some fixed, that define access to the Axeda Platform for an external application. The ExternalCredential Domain object, a Web Services V2 Bridge class, and a REST service support the External Credential feature. The methods of the bridge class and the operations of the REST service enable you to:

- Create a single Credential
- Retrieve a single Credential by exact name
- Find a list of Credentials by (supporting the V2 wildcard pattern) the Credential name, the Credential user name, or the Credential endpoint
- Modify a single Credential, or a list of Credentials by name
- Delete a single Credential, or a list of Credentials by name

Privileges Required

Privileges for viewing, creating, editing, and deleting External Credentials are broken up into a single Global Privilege, and a set of User Group Privileges. The Global Privilege provides access controls access to the SDK for External Credentials. The User Group privileges control the user's ability to associate user groups with an External Credential as well as the granular control for access to operations on a given credential.

The Global Privilege is called "Configuration - External Credential - Manage." This privilege allows a user to use the Axeda Platform API for External Credentials. Any User who is a member of a User Group that has been granted this Privilege can create, retrieve, update, and delete a credential.

User Groups are associated with credentials to provide granular access control. Two lists of groups are associated with each credential, "View" and "Modify". Members of the User Groups in the View list can read the contents of the credential, and can find this credential in searches. Members of the User Groups in the Modify list can modify and delete the credential. Membership in this list results in the same visibility as membership in the "View" list.

When adding User Groups to credentials, keep the following caveats in mind:

- A User cannot associate a User Group with a credential if that User cannot see that User Group.
- The only limit to the number of User Groups that you can associate with a given credential is the number of User Groups that you can see in the Axeda Platform. Otherwise, no limit is imposed.
- If a given credential has no associated User Groups, only the user who created it can access it.

Methods

The following table provides a brief overview of the available ExternalCredentialBridge methods, along with available REST calls. For a full description of ExternalCredentialBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
create	Creates a new ExternalCredential object.	PUT

		<server>/services/v2/rest/externalCre **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/externalCre
delete	Deletes a ExternalCredential object.	DELETE <server>/services/v2/rest/externalCredential/id/{id}
find (by criteria)	Finds ExternalCredential objects based on search	POST <server>/services/v2/rest/externalCredential/find
find (by IDs)	Finds ExternalCredential objects based on a list of	POST <server>/services/v2/rest/externalCredential/findByIds
find (by alternate ID)	Finds a ExternalCredential object based on the alternate identifier.	GET <server>/services/v2/rest/externalCredential/id/{id}
findById	Finds a ExternalCredential object based on its platform identifier.	GET <server>/services/v2/rest/externalCredential/id/{id}
findOne	Returns the first ExternalCredential object found that meets specified search criteria.	POST <server>/services/v2/rest/externalCredential/findOne
toString	Generates a string representing a specified ExternalCredential object.	N/A
update	Updates an existing ExternalCredential object.	POST <server>/services/v2/rest/externalCredential/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations:

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/externalCredential/bulk/create
bulkDelete	POST <server>/services/v2/rest/externalCredential/bulk/delete
bulkSave	POST <server>/services/v2/rest/externalCredential/bulk/save
bulkUpdate	POST <server>/services/v2/rest/externalCredential/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExternalCredentialBridge.html>

Create an ExternalCredential

Endpoint	/services/v2/rest/externalCredential
Verb(s)	PUT
Input Object	ExternalCredential
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalCredential
  <name>credential_1367257458628</name>
  <username>username</username>
  <password>password1</password>
  <domain>domain</domain>
  <apiKey>apiKey</apiKey>
  <apiSecret>apiSecret</apiSecret>
  <endpoint>http://endpoint</endpoint>
  <parameters>
    <namedValue>
      <name>name1</name>
      <value>value1</value>
    </namedValue>
    <namedValue>
      <name>name2</name>
      <value>value2</value>
    </namedValue>
  </parameters>
  <userGroupsWithViewPermission>
    <userGroup systemId="210"/>
  </userGroupsWithViewPermission>
  <userGroupsWithEditPermission>
    <userGroup systemId="211"/>
    <userGroup systemId="212"/>
  </userGroupsWithEditPermission>
</ExternalCredential>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded/>
  <v2:failures>
    <v2:failure code="OBJECT_NOT_FOUND">
      <v2:ref>credential_1367257458628</v2:ref>
      <v2:message>The ${target} was not found or permission
denied. user group id=210</v2:message>
      <v2:sourceOfFailure>userGroup</v2:sourceOfFailure>
    </v2:failure>
  </v2:failures>
</v2:ExecutionResult>
```

Update an

ExternalCredent

Endpoint	/services/v2/rest/externalCredential/id/96
Verb(s)	POST
Input Object	ExternalCredential
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalCredential
xmlns="http://www.axeda.com/services/v2" systemId="96">
  <name>changed_1367257460802</name>
  <username>changed_username</username>
  <password>changed_password</password>
  <domain>changed_domain</domain>
  <apiKey>changed_apiKey</apiKey>
  <apiSecret>changed_apiSecret</apiSecret>
  <endpoint>changed_http://endpoint</endpoint>
  <parameters>
    <namedValue>
      <name>changed_name1_1367257460797null</name>
      <value>changed_value1</value>
    </namedValue>
    <namedValue>
      <name>changed_name2_1367257460797null</name>
      <value>changed_value2</value>
    </namedValue>
  </parameters>
  <userGroupsWithViewPermission>
    <userGroup systemId="210"/>
  </userGroupsWithViewPermission>
  <userGroupsWithEditPermission>
    <userGroup systemId="211"/>
    <userGroup systemId="212"/>
  </userGroupsWithEditPermission>
</ExternalCredential>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded/>
  <v2:failures>
    <v2:failure code="OBJECT_NOT_FOUND">
      <v2:ref>96</v2:ref>
      <v2:message>The ${target} was not found or permission
denied.</v2:message>
      <v2:sourceOfFailure>
    </v2:failure>
  </v2:failures>
</v2:ExecutionResult>
```

Delete an ExternalCredent ial

Endpoint	/services/v2/rest/externalCredential/id/97
Verb(s)	DELETE
Input Object	

Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded/>
  <v2:failures>
    <v2:failure code="OBJECT_NOT_FOUND">
      <v2:ref>97</v2:ref>
      <v2:message>The ${target} was not found or permission
denied.</v2:message>
      <v2:sourceOfFailure>
    </v2:failure>
  </v2:failures>
</v2:ExecutionResult>
```

Save an ExternalCredential

Endpoint	/services/v2/rest/externalCredential
Verb(s)	POST
Input Object	ExternalCredential
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalCredential xmlns="http://www.axeda.com/services/v2"
systemId="92">
  <name>changed_1367257459373</name>
  <username>changed_username</username>
  <password>changed_password</password>
  <domain>changed_domain</domain>
  <apiKey>changed_apiKey</apiKey>
  <apiSecret>changed_apiSecret</apiSecret>
  <endpoint>changed_http://endpoint</endpoint>
  <parameters>
    <namedValue>
      <name>changed_name1_1367257459370null</name>
      <value>changed_value1</value>
    </namedValue>
    <namedValue>
      <name>changed_name2_1367257459370null</name>
      <value>changed_value2</value>
    </namedValue>
  </parameters>
  <userGroupsWithViewPermission>
    <userGroup systemId="210"/>
  </userGroupsWithViewPermission>
  <userGroupsWithEditPermission>
    <userGroup systemId="211"/>
    <userGroup systemId="212"/>
  </userGroupsWithEditPermission>
</ExternalCredential>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded/>
```

```
  <v2:failures>
    <v2:failure code="OBJECT_NOT_FOUND">
      <v2:ref>92</v2:ref>
      <v2:message>The ${target} was not found or permission
denied.</v2:message>
      <v2:sourceOfFailure>
    </v2:failure>
  </v2:failures>
</v2:ExecutionResult>
```

Find One External Credential

Endpoint	/services/v2/rest/externalCredential/findOne
Verb(s)	POST
Input	ExternalCredentialCriteria
Output Object	ExternalCredential

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalCredentialCriteria
  <name>credential_13675912121031367591212102-*</name>
  <username>*user*</username>
  <endpoint>*endpoint*</endpoint>
  <domain>*domain*</domain>
</ExternalCredentialCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExternalCredential xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="credential_13675912121031367591212102-1" systemId="76"
label="credential_13675912121031367591212102-1"
detail="http://endpoint" restUrl="http://jtelarico-
  <v2:name>credential_13675912121031367591212102-1</v2:name>
  <v2:username>username</v2:username>
  <v2:password>password</v2:password>
  <v2:domain>domain</v2:domain>
  <v2:apiKey>apiKey</v2:apiKey>
  <v2:apiSecret>apiSecret</v2:apiSecret>
  <v2:endpoint>http://endpoint</v2:endpoint>
  <v2:parameters>
    <v2:namedValue>
      <v2:name>name1_13675912121031367591212102-1</v2:name>
      <v2:value>value1</v2:value>
    </v2:namedValue>
    <v2:namedValue>
      <v2:name>name2_13675912121031367591212102-1</v2:name>
      <v2:value>value2</v2:value>
    </v2:namedValue>
  </v2:parameters>
  <v2:userGroupsWithViewPermission/>
  <v2:userGroupsWithEditPermission/>
  <v2:createdBy id="admin" systemId="1" label="admin admin"
detail="" restUrl="http://jtelarico-
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ExternalCredentialBridge.html>

FileInfoBridge

FileInfo objects contain metadata about a file. In order to upload a file to the Axeda Platform, it must be defined in a FileInfo object.

In order to create a FileInfo object, a file name and file size (or MD5) are required. A number of optional fields enable you to set FileInfo properties to facilitate searching, package deployment, file locking, and access privileges.

This section provides an overview of creating and manipulating FileInfo objects through the Axeda API using the FileInfoBridge. FileInfoBridge methods can also be accessed via REST calls over HTTP.

The FileInfoBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for FileInfo objects. Other methods enable you to get file data and check file upload progress.

FileInfo objects can contain properties such as file name, file size, description, available date, expiration date, and upload date. For a full description of FileInfo methods and properties, see the Axeda API Specification.

Methods

The following table provides a brief overview of the available FileInfoBridge methods, along with available REST calls. For a full description of FileInfoBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
create	Creates a new FileInfo object.	PUT <server>/services/v2/rest/file
delete (by file ID)	Deletes a FileInfo object based on ID.	DELETE <server>/services/v2/rest/file/{id}
delete (by name)	Deletes a FileInfo object based on file name.	POST <server>/services/v2/rest/file/delet
find (by criteria)	Finds FileInfo objects based on search criteria.	GET
find (by IDs)	Finds FileInfo objects based on a list of identifiers.	POST
find (by alternate ID)	Finds a FileInfo object based on the alternate	GET
findById	Finds a FileInfo object based on its platform	GET
findOne	Returns the first FileInfo object found that meets specified search criteria.	N/A
generateAlternateId	Returns a unique alternate identifier for the specified File Upload Session.	N/A
getFileData	Gets file content data as a stream of bytes.	GET <server>/services/v2/rest/file/{id}/download
getFileUploadPro	Returns the FileUploadProgress for the specified	GET

gress		<server>/services/v2/rest/file/{id}/progress
saveOrUpdate	Uploads and stores or overwrites the FileInfo	PUT <server>/services/v2/rest/file/{id}/content
toString	Generates a string representing a specified FileInfo	N/A
update	Updates an existing FileInfo object.	POST

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/FileInfoBridge.html>

Create a FileInfo Object

```
import com.axeda.services.v2.*;

FileInfo file = new FileInfo();
file.setFilename("foo");
file.setFilesize(123l);

def result = bridges.getFileInfoBridge().create(file);
```

Create a FileInfo Object with Privileges

```
import com.axeda.sdk.v2.bridge.*;
import com.axeda.services.v2.*;

FileInfo file = new FileInfo();
file.setFilename("foo");
file.setFilesize(123l);

FileInfoPrivilege readPriv = new FileInfoPrivilege();
readPriv.setGroupName("test_user_group");
readPriv.setPrivilege("R"); //READ
file.getPrivileges().add(readPriv);

FileInfoPrivilege modifyPriv = new FileInfoPrivilege();
modifyPriv.setGroupName("test_user_group");
modifyPriv.setPrivilege("U"); //UPDATE
file.getPrivileges().add(modifyPriv);

FileInfoBridge b = bridges.getFileInfoBridge();

b.create(file);
```

Find a FileInfo Object by Criteria

```
import com.axeda.sdk.v2.bridge.*;
import com.axeda.services.v2.*;

FileInfoCriteria c = new FileInfoCriteria();
c.setFilelabel("bar");

FileInfoBridge b = bridges.getFileInfoBridge();

FindFileInfoResult result = b.find(c);
return result;
```

Update a FileInfo Object

```
import com.axeda.sdk.v2.bridge.*;
import com.axeda.services.v2.*;

```

```
4 FileInfoBridge b = bridges.getFileInfoBridge();
5
6 FileInfo file = b.find("607");
7 file.setLabel("foo");
8
9 b.update(file);
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/FileInfoBridge.html>

FileUploadSessionBridge

FileUploadSession objects are sessions used to manage file uploads. Files can only be uploaded when associated with an active File Upload Session. To create a File Upload Session, a user must either be an administrator, or be associated with a group that has the file upload privilege.

FileUploadSession objects represent sessions for managing file uploads via REST Web Services. When a File Upload Session is successfully created, the ExecutionResult returns a success message along with the complete FileUploadSession with all IDs populated.

These values can be stored so that the session can be accessed at a later time.

Files can only be uploaded when associated with a non-timed-out session. Generally speaking, FileInfo objects should be created along with creation of a FileUploadSession.

If an existing FileInfo object needs new file data uploaded, it must first be associated or updated with an active upload session ID, as follows: create a new upload session, find and update an existing FileInfo object with the new session ID and other metadata, and then upload the new file data. If the updateLock is set, it must be cleared in order to upload new data.

When upload sessions are created, they don't require any properties to be populated. It may be desirable to set the expirationDate if many files will be uploaded, and therefore may take more time than the configurable expiration. The default is 24 hours.

FileUploadSession Properties

FileUploadSession objects can contain properties such as name, expiration date, status, and version. For a full description of FileUploadSession methods and properties, see the Axeda API Specification.

File Upload Sessions can be created and manipulated using the FileUploadSessionBridge object in the Axeda API. FileUploadSessionBridge methods can also be accessed via REST calls over HTTP.

The FileUploadSessionBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for FileUploadSession objects. Update and Delete operations are currently not allowed. Delete functionality can be achieved by setting the expirationDate property for a File Upload Session.

Methods

The following table provides a brief overview of the available FileUploadSessionBridge methods, along with available REST calls. For a full description of FileUploadSessionBridge methods, see the Axeda API Specification.

Method	Description	REST Call
create	Creates a new File Upload Session.	PUT <server>/services/v2/rest/file/ses
delete **NOT ALLOWED**	Deletes a File Upload Session.	N/A

find (by criteria)	Finds File Upload Sessions based on search criteria.	GET <server>/services/v2/rest/file/session/list
find (by IDs)	Finds File Upload Sessions based on a list of identifiers.	GET <server>/services/v2/rest/file/session/list
find (by alternate ID)	Finds a File Upload Session based on the alternate	GET <server>/services/v2/rest/file/session/{id}
findById	Finds a File Upload Session based on its platform	GET <server>/services/v2/rest/file/session/{id}
findOne	Returns the first File Upload Session found that meets specified search criteria.	N/A
generateAlternateId	Returns a unique alternate identifier for the specified File Upload Session.	N/A
toString	Generates a string representing a specified File Upload	N/A
update **NOT ALLOWED**	Updates an existing File Upload Session.	N/A

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/FileUploadSessionBridge.html>

Create a FileUploadSession

```
1  import com.axeda.sdk.v2.bridge.*;
2  import com.axeda.services.v2.*;
3
4  FileInfo file = new FileInfo();
5  file.setFilename("foo");
6  file.setFilesize(123l);
7
8  FileInfoPrivilege readPriv = new FileInfoPrivilege();
9  readPriv.setGroupName("test_user_group");
10 readPriv.setPrivilege("R");
11 file.getPrivileges().add(readPriv);
12
13 FileUploadSession session = new FileUploadSession();
14 session.getFiles().add(file);
15
16 FileUploadSessionBridge b = bridges.getFileUploadSessionBridge();
17
18 b.create(session);
```

Find a FileUploadSession by Criteria

```
1  import com.axeda.sdk.v2.bridge.*;
2  import com.axeda.services.v2.*;
3
4  FileUploadSessionCriteria c = new FileUploadSessionCriteria();
5  c.setStatus("PENDING");
6
7  FileUploadSessionBridge b = bridges.getFileUploadSessionBridge();
8
9  FindFileUploadSessionResult result = b.find(c);
10 return result;
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/FileUploadSessionBridge.html>

InstallSettingsBridge

The InstallSettingsBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for the Platform Settings Installation layer.

The configuration settings in the platform are structured in layers. The important idea is that each layer will be overlayed on the previous layer to form a complete view of the settings. Any setting defined in a lower layer will overwrite the same setting defined in an upper layer, if present. The three layers defined in the platform are (from top to bottom, with the Default layer being the highest layer, and the User layer being the lowest layer):

The Default layer — This layer contains the default settings specified by Axeda, and these settings can have any path. These configurations can be read by the install layer and they can be overwritten by the install and user layers.

The Installation layer — This layer contains any settings set by the installation administrator. This layer can contain settings with any path under /settings. These settings will overwrite the settings already defined in the Default layer. The InstallSettingsBridge is used to manage the settings in this layer.

The User layer — This layer contains personalized settings for each user. Access to this layer is based on the currently logged in user. The User layer cannot read or write to /settings/system path, as this path is reserved for the Default and Installation layers. Except for this reserved path, the User layer can contain settings with any other path under /settings. These settings will overwrite the settings defined in the Installation and Default layers. The UserSettingsBridge is used to manage the settings in this layer.

Methods

The InstallSettingsBridge is used to manage the settings in the Installation layer.

The following table provides a brief overview of the InstallSettingsBridge methods, along with available REST calls. For a full description of InstallSettingsBridge methods, see the Axeda API Specification.

Method	Description	REST Call
create	Creates a new PlatformSettings object.	PUT <server>/services/v2/rest/settings/install
delete	Deletes a PlatformSettings object.	DELETE <server>/services/v2/rest/settings/install
find (String path)	Finds the PlatformSettings object containing all the settings that match the specified path.	GET <server>/services/v2/rest/settings/install
findById	Since the PlatformSettings object does not have a system id, and it	GET

	can solely be identified by the root path, this method does the same thing as find(String path).	<server>/services/v2/rest/settings/install
fromXML	Builds a PlatformSettings object from the specified XML string.	N/A
generateAlternateId	A PlatformSettings object can be identified by its rootPath.	N/A
insertSetting	Updates the specified PlatformSettings instance by inserting a setting with the specified path and value.	N/A
removeNode	Updates the PlatformSettings by removing all the settings matching the specified nodePath.	N/A
removeSetting	Removes the specified Setting from the specified PlatformSettings	N/A
toString	Generates a string representing a specified Expression Rule.	N/A
toXML	Generates a XML string representing the PlatformSettings object.	N/A
update	Updates/replaces all the existing settings under the specified PlatformSettings.rootPath.	PUT <server>/services/v2/rest/settings/install
updateSettin	Updates the value of the setting that belongs to the specified	N/A

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/InstallSettingsBridge.html>

LocationBridge

Locations, along with Organizations and Regions, can be used to organize Assets.

Beginning with Axeda SDK v2, a Location must be associated with an Organization, and vice-versa. Therefore, an Organization (and Name) are required when creating a Location object. In order to associate a Location with a Region, an Organization and its corresponding Location must first be created.

This section provides an overview of creating and updating Location objects through the Axeda API using the LocationBridge. LocationBridge methods can also be accessed via REST calls over HTTP.

The LocationBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for Lcation objects.

Methods

The following table provides a brief overview of the available LocationBridge methods, along with available REST calls. For a full description of LocationBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
create	Creates a new Location object.	PUT <server>/services/v2/rest/location **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/location
delete	Deletes a Location object.	DELETE <server>/services/v2/rest/location/id/{id
find (by criteria)	Finds FileInfo objects based on search criteria.	POST
find (by IDs)	Finds Location objects based on a list of	POST <server>/services/v2/rest/location/findB
find (by alternate ID)	Finds a Location object based on the alternate	GET
findById	Finds a Location object based on its platform	GET
findOne	Returns the first Location object found that meets specified search criteria.	POST <server>/services/v2/rest/location/findO
toString	Generates a string representing a specified Location object.	N/A
update	Updates an existing Location object.	POST

Bulk REST Calls

As mentioned previously, in the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection,

depending on how many results were found.

The following REST Calls can be used for bulk operations:

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/location/bulk/create
bulkDelete	POST <server>/services/v2/rest/location/bulk/delete
bulkSave	POST <server>/services/v2/rest/location/bulk/save
bulkUpdate	POST <server>/services/v2/rest/location/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/LocationBridge.html>

Create a Location and an Organization

```
// Create a Location and an Organization as a composite object.
//This example creates both the Location and Organization
simultaneously,
//and so does not have to call LocationBridge.create
Location bostonCityHall = new Location(name:"Boston City Hall",
region:northEasternById, line1:"One City Hall Square", city:"Boston",
state:"MA", postalCode:"02201", country:"USA")
Organization massGov = new Organization(name:"Massachusetts
Government", description:"www.mass.gov")
massGov.locations.add(bostonCityHall)
create = organizationBridge.create(massGov)
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/LocationBridge.html>

Create a Location

Endpoint	/services/v2/rest/location
Verb(s)	PUT
Input Object	Location
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Location xmlns="http://www.axeda.com/services/v2">
  <name>loc name 1367591348596</name>
  <organization systemId="79"/>
  <region systemId="81"/>
  <line1>line 1 1367591348596</line1>
  <line2>line 2 1367591348596</line2>
  <city>city name 1367591348596</city>
  <country>country1367591348596</country>
  <countryCode>TST</countryCode>
  <language>test</language>
</Location>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>name_13675913362870||loc name
1367591348596</v2:ref>
      <v2:id>227</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find a Location by Criteria

Endpoint	/services/v2/rest/location/find
Verb(s)	POST
Input Object	LocationCriteria
Output Object	FindLocationResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<LocationCriteria xmlns="http://www.axeda.com/services/v2"
pageSize="10">
```

```
  <name>*1367591344446*</name>
  <organizationId>79</organizationId>
</LocationCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindLocationResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="2">
  <v2:criteria pageSize="10">
    <v2:name>*1367591344446*</v2:name>
    <v2:organizationId>79</v2:organizationId>
  </v2:criteria>
  <v2:locations>
    <v2:location xsi:type="v2:Location" id="name_13675913362870
||name_1367591344446a" systemId="218" label="name_
1367591344446a" detail="name_13675913362870"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/218">
      <v2:name>name_1367591344446a</v2:name>
      <v2:organization id="name_13675913362870" systemId="79"
label="name_13675913362870" detail="description_13675913362870"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/79"/>

      <v2:line1>1367591344446a</v2:line1>
      <v2:line2>1367591344446a</v2:line2>
      <v2:city>1367591344446a</v2:city>
      <v2:state>1367591344446a</v2:state>
      <v2:postalCode>1367591344446a</v2:postalCode>
      <v2:country>1367591344446a</v2:country>
      <v2:countryCode>US</v2:countryCode>
      <v2:language>English</v2:language>
    </v2:location>
    <v2:location xsi:type="v2:Location" id="name_13675913362870
||name_1367591344446b" systemId="219" label="name_
1367591344446b" detail="name_13675913362870"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/219">
      <v2:name>name_1367591344446b</v2:name>
      <v2:organization id="name_13675913362870" systemId="79"
label="name_13675913362870" detail="description_13675913362870"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/79"/>

      <v2:line1>1367591344446b</v2:line1>
      <v2:line2>1367591344446b</v2:line2>
      <v2:city>1367591344446b</v2:city>
      <v2:state>1367591344446b</v2:state>
      <v2:postalCode>1367591344446b</v2:postalCode>
      <v2:country>1367591344446b</v2:country>
      <v2:countryCode>US</v2:countryCode>
      <v2:language>English</v2:language>
    </v2:location>
  </v2:locations>
</v2:FindLocationResult>
```

Find a Location by System ID

Endpoint	/services/v2/rest/location/id/231
Verb(s)	GET
Input Object	
Output Object	Location

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:Location xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="name_
13675913362870||name_1367591351764" systemId="231" label="name_
1367591351764" detail="name_13675913362870"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/231">
  <v2:name>name_1367591351764</v2:name>
  <v2:organization id="name_13675913362870" systemId="79"
label="name_13675913362870" detail="description_13675913362870"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/79"/>
  <v2:line1>1367591351764</v2:line1>
  <v2:line2>1367591351764</v2:line2>
  <v2:city>1367591351764</v2:city>
  <v2:state>1367591351764</v2:state>
  <v2:postalCode>1367591351764</v2:postalCode>
  <v2:country>1367591351764</v2:country>
  <v2:countryCode>US</v2:countryCode>
  <v2:language>English</v2:language>
</v2:Location>
```

Update a Location

Endpoint	/services/v2/rest/location/id/228
Verb(s)	POST
Input Object	
Output Object	

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Location xmlns="http://www.axeda.com/services/v2"
systemId="228">
  <name>loc name 1367591349365</name>
  <organization systemId="79"/>
  <region systemId="81"/>
  <line1>line 1 1367591349365</line1>
  <line2>line 2 1367591349365</line2>
  <city>city name 1367591349365</city>
  <country>country1367591349365</country>
  <countryCode>TST</countryCode>
  <language>test</language>
</Location>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
```

```
    <v2:success>
      <v2:ref>228</v2:ref>
      <v2:id>228</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete a Location

Endpoint	/services/v2/rest/location/id/230
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>230</v2:ref>
      <v2:id>230</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/LocationBridge.html>

ModelBridge

A single Model object can contain many assets of the same type and properties. For example, ModelXYZ may contain 50 identical connected, intelligent assets, all intended for the same functionality.

Models can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating models through the API using the ModelBridge.

The ModelBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for Model objects.

Methods

The following table provides a brief overview of the ModelBridge methods, along with available REST calls. For a full description of ModelBridge methods, see the Axeda API Specification.

Method	Description	REST Call
create	Creates a new model.	PUT <server>/services/v2/rest/model **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/model
delete	Deletes a model.	DELETE
find (by IDs)	Finds models based on a list of identifiers.	POST
find (by criteria)	Finds models based on search criteria.	POST <server>/services/v2/rest/model/find
find (by alternate ID)	Finds a model based on the alternate	GET
findById	Finds a model based on its platform identifier.	GET
findOne	Returns the first model found that meets specified search criteria.	POST
toString	Generates a string representing a specified	N/A
update	Updates an existing model.	POST

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using Model Collections.

Operation	REST Call

bulkCreate	POST <server>/services/v2/rest/model/bulk/create
bulkDelete	POST <server>/services/v2/rest/model/bulk/delete
bulkSave	POST <server>/services/v2/rest/model/bulk/save
bulkUpdate	POST <server>/services/v2/rest/model/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ModelBridge.html>

Wednesday, March 01, 2017 10:35 AM

Create a Model

```
import com.axeda.services.v2.*
Model model = new Model(modelNumber:"FooModel")
ExecutionResult executionResult = bridges.modelBridge.create(model)
```

Find a Model by Criteria

```
import com.axeda.services.v2.*
ModelCriteria modelCriteria = new ModelCriteria(modelNumber:"FooModel")

return bridges.modelBridge.find(modelCriteria))
```

Update a Model

```
   import com.axeda.services.v2.*
   Model model = bridges.modelBridge.find("FooModel")
   if(!model)
   {
     return 'the model was not found - create it first'
   }

   model.setDescription("i changed the model's description")
   ExecutionResult updateResult = bridges.modelBridge.update(model)
   if(!updateResult.successful)
   {
     //something bad happened!
     return updateResult
   }
 // find it back out and prove it's been updated
return bridges.modelBridge.find("FooModel")
```

Delete a Model

```
return bridges.modelBridge.delete(bridges.modelBridge.find("FooModel"))
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ModelBridge.html>

Create a Model

Endpoint	/services/v2/rest/model
Verb(s)	PUT
Input Object	Model
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Model xmlns="http://www.axeda.com/services/v2">
  <name>model name 1367256926285</name>
  <modelNumber>model no 1367256926285</modelNumber>
  <description>Description1367256926285</description>
  <type>STANDALONE</type>
  <missingStrategy>
    <pingMultiplier>10</pingMultiplier>
    <additionalFactor>
      <value>1000</value>
      <units>SECONDS</units>
    </additionalFactor>
  </missingStrategy>
</Model>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>model no 1367256926285</v2:ref>
      <v2:id>800</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find One Model

Endpoint	/services/v2/rest/model/findOne
Verb(s)	POST
Input Object	ModelCriteria
Output Object	Model

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ModelCriteria xmlns="http://www.axeda.com/services/v2">
```

```
  <modelNumber>*1367591383245*</modelNumber>
</ModelCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:Model xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="model
no 1367591383245a" systemId="90" label="standalone" detail="model
no 1367591383245a" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/90">

  <v2:name>model name 1367591383245a</v2:name>
  <v2:modelNumber>model no 1367591383245a</v2:modelNumber>
  <v2:description>Description1367591383245a</v2:description>

  <v2:autoRegisterAssets>false</v2:autoRegisterAssets>
  <v2:type>STANDALONE</v2:type>
  <v2:defaultAssetGroup id="/Root Asset Group/Default Model
Group/model no 1367591383245a Default Group" systemId="172"
label="model no 1367591383245a Default Group" detail="/Root Asset
Group/Default Model Group" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/assetGroup/172"/>
  <v2:dataItems/>
  <v2:modelAlarms/>
  <v2:properties/>
  <v2:gatewayParameters/>
  <v2:missingStrategy>
    <v2:pingMultiplier>10</v2:pingMultiplier>
    <v2:additionalFactor>
      <v2:value>1000</v2:value>
      <v2:units>SECONDS</v2:units>
    </v2:additionalFactor>
  </v2:missingStrategy>
  <v2:sourceModels/>
  <v2:targetModels/>
</v2:Model>
```

Find a Model by System ID

Endpoint	/services/v2/rest/model/id/89
Verb(s)	GET
Input Object	
Output Object	Model

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:Model xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="model
no 1367591381531" systemId="89" label="standalone" detail="model
no 1367591381531" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/model/89">

  <v2:name>model name 1367591381531</v2:name>
  <v2:modelNumber>model no 1367591381531</v2:modelNumber>
  <v2:description>Description1367591381531</v2:description>

  <v2:autoRegisterAssets>false</v2:autoRegisterAssets>
```

```
  <v2:type>STANDALONE</v2:type>
  <v2:defaultAssetGroup id="/Root Asset Group/Default Model
Group/model no 1367591381531 Default Group" systemId="171"
label="model no 1367591381531 Default Group" detail="/Root Asset
Group/Default Model Group" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/assetGroup/171"/>

  <v2:dataItems/>
  <v2:modelAlarms/>
  <v2:properties/>
  <v2:gatewayParameters/>
  <v2:missingStrategy>
    <v2:pingMultiplier>10</v2:pingMultiplier>
    <v2:additionalFactor>
      <v2:value>1000</v2:value>
      <v2:units>SECONDS</v2:units>
    </v2:additionalFactor>
  </v2:missingStrategy>
  <v2:sourceModels/>
  <v2:targetModels/>
</v2:Model>
```

Update a Model

Endpoint	/services/v2/rest/model/id/86
Verb(s)	POST
Input Object	
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Model xmlns="http://www.axeda.com/services/v2" systemId="86">

  <name>model name 1367591377984</name>
  <modelNumber>model no 1367591377031</modelNumber>
  <description>Description1367591377984</description>
  <type>STANDALONE</type>
  <defaultAssetGroup systemId="168"/>
  <missingStrategy>
    <pingMultiplier>10</pingMultiplier>
    <additionalFactor>
      <value>1000</value>
      <units>SECONDS</units>
    </additionalFactor>
  </missingStrategy>
</Model>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>86</v2:ref>
      <v2:id>86</v2:id>
    </v2:success>
```

```
    </v2:succeeded>
    <v2:failures/>
  </v2:ExecutionResult>
```

Delete a Model	
Endpoint	/services/v2/rest/model/id/88
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult
XML Response	

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  successful="true" totalCount="1">
5
6    <v2:succeeded>
7      <v2:success>
8        <v2:ref>88</v2:ref>
9        <v2:id>88</v2:id>
10     </v2:success>
     </v2:succeeded>
     <v2:failures/>
   </v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/ModelBridge.html>

OrganizationBridge

Organizations, along with Locations and Regions, can be used to organize Assets.

In the Axeda v2 SDK, a Location must be associated with an Organization, and vice-versa. An Organization must have at least one Location, and that Location cannot be shared with other Organizations. When creating an Organization object, a Location (and Name) are required (and optionally, a description can be included).

Organization objects can be created and updated through the Axeda API using the OrganizationBridge. OrganizationBridge methods can also be accessed via REST calls over HTTP.

The OrganizationBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for Organization objects.

Methods

The following table provides a brief overview of the available OrganizationBridge methods, along with available REST Calls. For a full description of OrganizationBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
create	Creates a new Organization object.	PUT **Note:** this same REST call as a POST invokes a Save operation: POST
delete	Deletes an Organization object.	DELETE <server>/services/v2/rest/organization/id/{id}
find (by criteria)	Finds Organization objects based on search	POST <server>/services/v2/rest/organizatio
find (by IDs)	Finds Organization objects based on a list of	POST <server>/services/v2/rest/organization/findByIds
find (by alternate ID)	Finds an Organization object based on the alternate identifier.	GET <server>/services/v2/rest/organization/id/{id}
findById	Finds an Organization object based on its platform identifier.	GET <server>/services/v2/rest/organization/id/{id}
findOne	Returns the first Organization object found that meets specified search criteria.	POST <server>/services/v2/rest/organization/findOne

toString	Generates a string representing a specified Organization object.	N/A
update	Updates an existing Organization object.	POST <server>/services/v2/rest/organization/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations:

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/organization/bulk/create
bulkDelete	POST <server>/services/v2/rest/organization/bulk/delete
bulkSave	POST <server>/services/v2/rest/organization/bulk/save
bulkUpdate	POST <server>/services/v2/rest/organization/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/OrganizationBridge.html>

Create an Organization

```
/**
 * An Organization consists of a name, an optional description and at
least one mandatory Location.
 *
 * Example:
 */

import static com.axeda.sdk.v2.dsl.Bridges.*

import com.axeda.services.v2.ExecutionResult;
import com.axeda.services.v2.FindLocationResult;
import com.axeda.services.v2.FindOrganizationResult;
import com.axeda.services.v2.Location
import com.axeda.services.v2.LocationCriteria;
import com.axeda.services.v2.Organization;
import com.axeda.services.v2.OrganizationCriteria;

def didItWork = { it ->
  assert it.successful, it.getFailures()[0].getMessage().replaceAll(/
\$\{target\}/, it.getFailures()[0].getSourceOfFailure())

}

def response = "success"

try {
  ExecutionResult result
  Organization acme = new Organization(name:"Acme Corporation",
description:"Wile E. Coyote's preferred vendor.")
  Location acmeMonumentValley = new Location(name:"Acme Monument Valley",
line1:"P.O. Box A1", city:"Monument Valley", state:"UT",
postalCode:"84536")
  acme.locations.add(acmeMonumentValley) // Organization requires a
Location

  Organization petco = new Organization(name:"Petco",
description:"Roadrunner's preferred vendor.")
  Location petcoFlagstaff = new Location(name:"Petco Flagstaff",
line1:"5047 East Marketplace Drive", city:"Flagstaff", state:"AZ",
postalCode:"86004")
  petco.locations.add(petcoFlagstaff)

  // CREATE
  result = organizationBridge.create(acme)
  didItWork(result)
  result = organizationBridge.create(petco)
  didItWork(result)
```

Find an Organization by Criteria

```
OrganizationCriteria orgCriteria = new OrganizationCriteria(name:"Acme
```

```
2 Corporation")
  FindOrganizationResult acmeAmongOthersByCriteria =
  organizationBridge.find(orgCriteria)
```

Find an Organization by System ID

```
1 OrganizationCriteria orgCriteria = new OrganizationCriteria(name:"Acme
2 Corporation")
  Organization acmeById = organizationBridge.findById(oneAcmeByCriteria.id)
```

Update an Organization

```
1 def oldDesc = acmeById.description
2 acmeById.description = oldDesc + " CHANGED"
3 organizationBridge.update(acmeById)
4 acmeById = organizationBridge.findById(acmeById.id)
5 assert acmeById.description.equals(oldDesc + " CHANGED")
```

Delete an Organization

```
1 organizationBridge.delete(acmeById)
2 organizationBridge.delete(petcoByAltId)
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/OrganizationBridge.html>

Create an Organization

Endpoint	/services/v2/rest/organization
Verb(s)	PUT
Input Object	Organization
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Organization xmlns="http://www.axeda.com/services/v2">
  <name>organizationName1367591415756</name>
  <description>
organizationDescription1367591415756</description>
  <locations>
    <location xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="Location">

      <name>locationNameA_1367591415756</name>
      <line1>line1A_1367591415756</line1>
      <line2>line1A_1367591415756</line2>
      <city>cityNameA_1367591415756</city>
      <country>countryA_1367591415756</country>
      <countryCode>TST</countryCode>
      <language>test</language>
    </location>
    <location xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="Location">

      <name>locationNameB_1367591415756</name>
      <line1>line1B_1367591415756</line1>
      <line2>line1B_1367591415756</line2>
      <city>cityNameB_1367591415756</city>
      <country>countryB_1367591415756</country>
      <countryCode>TST</countryCode>
      <language>test</language>
    </location>
  </locations>
</Organization>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>organizationName1367591415756</v2:ref>
      <v2:id>96</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find an Organization by Criteria	
Endpoint	/services/v2/rest/organization/find
Verb(s)	POST
Input Object	OrganizationCriteria
Output Object	FindOrganizationResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<OrganizationCriteria
xmlns="http://www.axeda.com/services/v2">
  <name>*1367591409521</name>
</OrganizationCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindOrganizationResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="3">
  <v2:criteria>
    <v2:name>*1367591409521</v2:name>
  </v2:criteria>
  <v2:organizations>
    <v2:organization xsi:type="v2:Organization"
id="organizationName01367591409521" systemId="81"
label="organizationName01367591409521"
detail="organizationDescription01367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/81">
      <v2:name>organizationName01367591409521</v2:name>
      <v2:description>
organizationDescription01367591409521</v2:description>
      <v2:locations>
        <v2:location id="organizationName01367591409521
||locationNameA_01367591409521" systemId="236"
label="locationNameA_01367591409521"
detail="organizationName01367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/236"/>
        <v2:location id="organizationName01367591409521
||locationNameB_01367591409521" systemId="237"
label="locationNameB_01367591409521"
detail="organizationName01367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/237"/>
      </v2:locations>
    </v2:organization>
    <v2:organization xsi:type="v2:Organization"
id="organizationName11367591409521" systemId="82"
label="organizationName11367591409521"
detail="organizationDescription11367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/82">
      <v2:name>organizationName11367591409521</v2:name>
      <v2:description>
organizationDescription11367591409521</v2:description>
      <v2:locations>
```

```
        <v2:location id="organizationName11367591409521
||locationNameA_11367591409521" systemId="238"
label="locationNameA_11367591409521"
detail="organizationName11367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/238"/>
        <v2:location id="organizationName11367591409521
||locationNameB_11367591409521" systemId="239"
label="locationNameB_11367591409521"
detail="organizationName11367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/239"/>
      </v2:locations>
    </v2:organization>
    <v2:organization xsi:type="v2:Organization"
id="organizationName21367591409521" systemId="83"
label="organizationName21367591409521"
detail="organizationDescription21367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/organization/83">
      <v2:name>organizationName21367591409521</v2:name>
      <v2:description>
organizationDescription21367591409521</v2:description>
      <v2:locations>
        <v2:location id="organizationName21367591409521
||locationNameA_21367591409521" systemId="240"
label="locationNameA_21367591409521"
detail="organizationName21367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/240"/>
        <v2:location id="organizationName21367591409521
||locationNameB_21367591409521" systemId="241"
label="locationNameB_21367591409521"
detail="organizationName21367591409521"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/241"/>
      </v2:locations>
    </v2:organization>
  </v2:organizations>
</v2:FindOrganizationResult>
```

Save an Organization

Endpoint	/services/v2/rest/organization
Verb(s)	POST
Input Object	Organization
Output Object	ExecutionResult

XML Request

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Organization xmlns="http://www.axeda.com/services/v2"
3 systemId="98">
4   <name>changedorganizationName1367591418713</name>
5   <description>
6 changedorganizationDescription1367591418713</description>
7
8   <locations>
```

```
    <location xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="Location">

      <name>locationNameA_1367591418713</name>
      <line1>line1A_1367591418713</line1>
      <line2>line1A_1367591418713</line2>
      <city>cityNameA_1367591418713</city>
      <country>countryA_1367591418713</country>
      <countryCode>TST</countryCode>
      <language>test</language>
    </location>
    <location xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="Location">

      <name>locationNameB_1367591418713</name>
      <line1>line1B_1367591418713</line1>
      <line2>line1B_1367591418713</line2>
      <city>cityNameB_1367591418713</city>
      <country>countryB_1367591418713</country>
      <countryCode>TST</countryCode>
      <language>test</language>
    </location>
  </locations>
</Organization>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>98</v2:ref>
      <v2:id>98</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Update an Organization

Endpoint	/services/v2/rest/organization/id/97
Verb(s)	POST
Input Object	Organization
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Organization xmlns="http://www.axeda.com/services/v2"
systemId="97">
  <name>changedorganizationName1367591417224</name>
  <description>
changedorganizationDescription1367591417224</description>

  <locations>
    <location xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="Location">

      <name>locationNameA_1367591417224</name>
      <line1>line1A_1367591417224</line1>
```

```
      <line2>line1A_1367591417224</line2>
      <city>cityNameA_1367591417224</city>
      <country>countryA_1367591417224</country>
      <countryCode>TST</countryCode>
      <language>test</language>
    </location>
    <location xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="Location">

      <name>locationNameB_1367591417224</name>
      <line1>line1B_1367591417224</line1>
      <line2>line1B_1367591417224</line2>
      <city>cityNameB_1367591417224</city>
      <country>countryB_1367591417224</country>
      <countryCode>TST</countryCode>
      <language>test</language>
    </location>
  </locations>
</Organization>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>97</v2:ref>
      <v2:id>97</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete an Organization

Endpoint	/services/v2/rest/organization/id/99
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">
  <v2:succeeded>
    <v2:success>
      <v2:ref>99</v2:ref>
      <v2:id>99</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/OrganizationBridge.html>

RegionBridge

Regions, along with Locations and Organizations, can be used to organize Assets.

Regions are hierarchical in nature with no limit to depth, although two to three levels of hierarchy should suffice for most applications. Regions can have parent-child relationships with other Regions, and can also be associated with Locations. A Location must be associated with an Organization before it can be associated with a Region.

Region objects can be created and updated through the Axeda API using the RegionBridge. RegionBridge methods can also be accessed via REST calls over HTTP.

The RegionBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for Region objects.

Methods

The following table provides a brief overview of the available RegionBridge methods, along with available REST Calls. For a full description of RegionBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
create	Creates a new Region object.	PUT <server>/services/v2/rest/region **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/region
delete	Deletes a Region object.	DELETE
find (by criteria)	Finds Region objects based on search criteria.	POST
find (by IDs)	Finds Region objects based on a list of identifiers.	POST
find (by alternate ID)	Finds a Region object based on the alternate	GET
findById	Finds a Region object based on its platform	GET
findOne	Returns the first Region object found that meets specified search criteria.	POST
toString	Generates a string representing a specified	N/A
update	Updates an existing Region object.	POST

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many

results were found.

The following REST Calls can be used for bulk operations:

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/region/bulk/create
bulkDelete	POST <server>/services/v2/rest/region/bulk/delete
bulkSave	POST <server>/services/v2/rest/region/bulk/save
bulkUpdate	POST <server>/services/v2/rest/region/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/RegionBridge.html>

RegionBridge CRUD methods

```
 /**
 * Regions are hierarchical in nature with no limit to depth, although two
to
 * three levels of hierarchy should suffice for most applications.
 *
 * This example starts off by demonstrating the parent/child relationships
of Regions
 * and their possible association with Locations.
 *
 * Beginning with with SDKv2, a Location must be associated with an
Organization
 * and vice-versa, so in order to demonstrate associating a Location with
a Region,
 * an Organization and its corresponding Location are created first.
 */
import static com.axeda.sdk.v2.dsl.Bridges.*

import com.axeda.sdk.v2.transformers.RegionTransformer;
import com.axeda.services.v2.ExecutionResult;
import com.axeda.services.v2.FindRegionResult;
import com.axeda.services.v2.Location;
import com.axeda.services.v2.LocationCriteria;
import com.axeda.services.v2.Organization;
import com.axeda.services.v2.OrganizationCriteria;
import com.axeda.services.v2.Region;
import com.axeda.services.v2.RegionCriteria;
import com.axeda.services.v2.RegionReference;

def response = "success"

def didItWork = { it ->
  assert it.successful, it.getFailures()[0].getMessage().replaceAll(/
\$\{target\}/, it.getFailures()[0].getSourceOfFailure())

}

try {
  // CREATE a parent Region with two children
  def unitedStates = "United States"
  def northEastern = "Northeastern"
  def southern = "Southern"

  Region unitedStatesRegion = new Region(name:unitedStates)
  Region northEasternRegion = new Region(name:northEastern,
parent:unitedStatesRegion)
  Region southernRegion = new Region(name:southern,
parent:unitedStatesRegion)

  ExecutionResult create = regionBridge.create(unitedStatesRegion)
  didItWork(create)
  def unitedStatesRegionId = create.succeeded[0].id

  create = regionBridge.create(northEasternRegion)
```

```
  didItWork(create)
  def northEasternRegionId = create.succeeded[0].id

  create = regionBridge.create(southernRegion)
  didItWork(create)
  def southernRegionId = create.succeeded[0].id

  // READ and verify parent / child relationship
  Region unitedStatesById = regionBridge.findById(unitedStatesRegionId)

  Region northEasternById = regionBridge.findById(northEasternRegionId)

  Region southernById = regionBridge.findById(southernRegionId)

  // Note that the parent region got associated with its children by the
platform implicitly when the children were created.
  assert unitedStatesById.children[0].id == northEasternById.id, "United
States Region should know its child is Northeastern Region"
  assert unitedStatesById.children[1].id == southernById.id, "United
States Region should know its child is Southern Region"
  assert northEasternById.parent.id == unitedStatesById.id, "Northeastern
Region should know its parent is United States Region"
  assert southernById.parent.id == unitedStatesById.id, "Southern Region
should know its parent is United States Region"

  // Now let's UPDATE a region, adding a description
  unitedStatesById.description = "A federal constitutional republic
comprising fifty states and a federal district."
  regionBridge.update(unitedStatesById)
  Region unitedStatesByName = regionBridge.find(unitedStatesById.name) //
Find (READ) by name this time.  Because alternateId for Region is name,
this works.
  assert unitedStatesByName.description == unitedStatesById.description,
"Description did not get updated"

  // Regions can be include specific locations, but Locations are tied to
Organizations and vice versa and must be created as a composite object.

  // So we have to create an Organization with a Location first.
  Location bostonCityHall = new Location(name:"Boston City Hall",
region:northEasternById, line1:"One City Hall Square", city:"Boston",
state:"MA", postalCode:"02201", country:"USA")
  Organization massGov = new Organization(name:"Massachusetts Government",
description:"www.mass.gov")
  massGov.locations.add(bostonCityHall)
  create = organizationBridge.create(massGov)
  didItWork(create)

  // Verify the Organization with Location can be READ.
  Organization massGovByName = organizationBridge.find("Massachusetts
Government")
  Location bostonByName = locationBridge.findOne(new
LocationCriteria(name:"Boston City Hall"))
  assert massGovByName != null, "Could not find Massachusetts Government
by name"
  assert bostonByName != null, "Could not find Boston City Hall by name"

  assert massGovByName.locations[0].id == bostonByName.id, "Massachusetts
Government Organization does not have the Boston City Hall Location"
```

```
  // Finally, add the Location of the newly created Organization to a
Region
  Region northEastByCriteria = regionBridge.findOne(new
RegionCriteria(name:northEastern))  // Another way to find (READ) using
RegionCriteria
  northEastByCriteria.locations.add(bostonByName)
  regionBridge.update(northEastByCriteria)
  northEastByCriteria = regionBridge.findById(northEastByCriteria.id)

  assert northEastByCriteria.locations.get(0).id == bostonByName.id,
"Boston City Hall was not successfully added to the Northeast region"


  // Now let's clean up with some DELETEs
  regionBridge.delete(unitedStatesById)
  regionBridge.delete(southernById)
  regionBridge.delete(northEasternById)
  organizationBridge.delete(massGovByName)
} catch (Exception e) {
  response = e.getLocalizedMessage()
} catch (AssertionError e) {
  response = e.getLocalizedMessage()
}

return response
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/RegionBridge.html>

Create a Region

Endpoint	/services/v2/rest/region
Verb(s)	PUT
Input Object	Region
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Region xmlns="http://www.axeda.com/services/v2"
label="labelcom.axeda.UtilsTestgetUniqueString1367838108479
_CREATE">
  <name>namecom.axeda.UtilsTestgetUniqueString1367838108479
_CREATE</name>

  <description>
desccom.axeda.UtilsTestgetUniqueString1367838108479
_CREATE</description>
  <parent systemId="30"/>
  <children>
    <region systemId="31"/>
    <region systemId="32"/>
  </children>
  <locations>
    <location systemId="39"/>
  </locations>
</Region>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>
namecom.axeda.UtilsTestgetUniqueString1367838108479
_CREATE</v2:ref>
        <v2:id>33</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find a Region by Criteria

Endpoint	/services/v2/rest/region/find
Verb(s)	POST
Input	RegionCriteria

Object

Output Object: FindRegionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<RegionCriteria xmlns="http://www.axeda.com/services/v2"
pageSize="5" pageNumber="1">
  <description>*com.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_*</description>
</RegionCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindRegionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
totalCount="2">

  <v2:criteria pageSize="5" pageNumber="1">
    <v2:description>*com.axeda.UtilsTestgetUniqueString1367838097
796_FIND_BY_CRITERIA_*</v2:description>
  </v2:criteria>
  <v2:regions>
    <v2:region xsi:type="v2:Region"
id="namecom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_1" systemId="12"
label="namecom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_1"
detail="desccom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_1" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/region/12">
      <v2:name>
namecom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_1</v2:name>
      <v2:description>
desccom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_1</v2:description>
      <v2:parent
id="namecom.axeda.UtilsTestgetUniqueString1367838097796_01"
systemId="9"
label="namecom.axeda.UtilsTestgetUniqueString1367838097796_01"
detail="desccom.axeda.UtilsTestgetUniqueString1367838097796_01"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/region/9"/>

      <v2:children/>
      <v2:locations>
        <v2:location
id="name_PS_com.axeda.UtilsTestgetUniqueString13678380977960||loc
name com.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_1" systemId="18" label="loc name
com.axeda.UtilsTestgetUniqueString1367838097796_FIND_BY_CRITERIA_
1"
detail="name_PS_com.axeda.UtilsTestgetUniqueString13678380977960"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/18"/>

      </v2:locations>
    </v2:region>
    <v2:region xsi:type="v2:Region"
id="namecom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_2" systemId="13"
```

```
label="namecom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_2"
detail="desccom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_2" restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/region/13">
      <v2:name>
namecom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_2</v2:name>
      <v2:description>
desccom.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_2</v2:description>
      <v2:children>
        <v2:region
id="namecom.axeda.UtilsTestgetUniqueString1367838097796_02"
systemId="10"
label="namecom.axeda.UtilsTestgetUniqueString1367838097796_02"
detail="desccom.axeda.UtilsTestgetUniqueString1367838097796_02"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/region/10"/>

        <v2:region
id="namecom.axeda.UtilsTestgetUniqueString1367838097796_03"
systemId="11"
label="namecom.axeda.UtilsTestgetUniqueString1367838097796_03"
detail="desccom.axeda.UtilsTestgetUniqueString1367838097796_03"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/region/11"/>

      </v2:children>
      <v2:locations>
        <v2:location
id="name_PS_com.axeda.UtilsTestgetUniqueString13678380977960||loc
name com.axeda.UtilsTestgetUniqueString1367838097796
_FIND_BY_CRITERIA_2" systemId="19" label="loc name
com.axeda.UtilsTestgetUniqueString1367838097796_FIND_BY_CRITERIA_
2"
detail="name_PS_com.axeda.UtilsTestgetUniqueString13678380977960"
restUrl="http://jtelarico-
dt7.axeda.com:8080/services/v2/rest/location/19"/>

      </v2:locations>
    </v2:region>
  </v2:regions>
</v2:FindRegionResult>
```

Update a Region	
Endpoint	/services/v2/rest/region/id/37
Verb(s)	POST
Input Object	Region
Output Object	ExecutionResult
XML Request	

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Region xmlns="http://www.axeda.com/services/v2" systemId="37"
3 label="labelcom.axeda.UtilsTestgetUniqueString1367838111096
4 _UPDATE">
5
```

```
  <name>
UPDATED_NAME_com.axeda.UtilsTestgetUniqueString1367838111096</na
me>
  <description>
desccom.axeda.UtilsTestgetUniqueString1367838111096
_UPDATE</description>
  <children>
    <region systemId="35"/>
    <region systemId="36"/>
  </children>
  <locations>
    <location systemId="43"/>
  </locations>
</Region>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>37</v2:ref>
      <v2:id>37</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Save a Region

Endpoint	/services/v2/rest/region
Verb(s)	POST
Input Object	Region
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Region xmlns="http://www.axeda.com/services/v2"
label="labelcom.axeda.UtilsTestgetUniqueString1367838113773
_SAVE">
  <name>namecom.axeda.UtilsTestgetUniqueString1367838113773
_SAVE</name>

  <description>
desccom.axeda.UtilsTestgetUniqueString1367838113773
_SAVE</description>
  <parent systemId="38"/>
  <children>
    <region systemId="39"/>
    <region systemId="40"/>
  </children>
  <locations>
    <location systemId="47"/>
  </locations>
</Region>
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

Response

```
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>namecom.axeda.UtilsTestgetUniqueString1367838113773
_SAVE</v2:ref>

      <v2:id>41</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete a Region

Endpoint	/services/v2/rest/region/id/45
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
successful="true" totalCount="1">

  <v2:succeeded>
    <v2:success>
      <v2:ref>45</v2:ref>
      <v2:id>45</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/RegionBridge.html>

RemoteSessionBridge

Remote Sessions enable service personnel to access and communicate with devices hidden behind corporate firewalls, when those devices do not have a graphical desktop interface. Remote Sessions also allow for sharing of an Axeda Agent device's desktop. There are four types of remote sessions: Remote Terminal, Remote Application, Remote Browser, and Remote Desktop.

A Remote Session can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating Remote Sessions through the API using the RemoteSessionBridge object, and also lists the available REST calls.

The RemoteSessionBridge object provides Create and Find operations for RemoteSession objects. You can also use the RemoteSessionBridge to start and end Remote Sessions, to find Remote Sessions, and to find the Remote Interfaces associated with a specified Asset.

Methods

The following table provides a brief overview of the RemoteSessionBridge methods, along with available REST calls. For a full description of RemoteSessionBridge methods, see the Axeda API Specification.

Method	Description	REST Call
createSession	Creates a Remote Session based on a set of input parameters.	PUT
delete	Deletes a Remote Session.	DELETE <server>/services/v2/rest/remoteSession
deleteHistory	Deletes a Remote Session.	DELETE <server>/services/v2/rest/remoteSession
endSession	Attempts to end the Remote Session referenced by the specified sessionId.	POST <server>/services/v2/rest/remoteSession/id/{id}/end
find (by criteria)	Finds Remote Sessions based on search criteria.	POST <server>/services/v2/rest/remoteSession
find (by IDs)	Finds Remote Sessions based on a list of IDs.	POST <server>/services/v2/rest/remoteSession/findByIds
findById	Finds a RemoteSession object based on its platform identifier (systemId).	GET <server>/services/v2/rest/remoteSession
find (by alternate ID)	Finds a RemoteSession object based on the alternate identifier.	GET <server>/services/v2/rest/remoteSession
findOne	Returns the first Remote Session found that meets specified search criteria.	POST <server>/services/v2/rest/remoteSession/findOne
findInterface	Finds a RemoteInterface based on the specified Platform identifier.	GET <server>/services/v2/rest/remoteSession/interface/id/{id}

findInterfaces	Finds the active RemoteInterfaces exposed by an Asset based on the specified Asset reference.	GET <server>/services/v2/rest/remoteSession/findInterfacesByAssetId -or- GET <server>/services/v2/rest/remoteSession/findInterfacesByAssetAlternateId
getSessionStatus	Returns the status of the Remote Session referenced by the specified sessionId.	N/A
isUserValidForSession	Checks to see if the given user name is valid for the specified Remote Session.	POST <server>/services/v2/rest/remoteSession/id/{id}/isUserValid
joinSession	Enables the current user to join the specified Telnet Remote Session.	POST <server>/services/v2/rest/remoteSession/id/{id}/join
leaveSession	Enables the current user to leave the specified Telnet Remote Session.	POST <server>/services/v2/rest/remoteSession/id/{id}/leave
mergeSession	Creates a new Remote Session and merges it with a specified existing Remote Session.	POST <server>/services/v2/rest/remoteSession/id/{id}/merge
toString	Generates a string representing a specified Remote Session.	N/A

Bulk REST Calls

For Remote Session, only the bulk delete operation is supported. The following REST Call can be used for bulk delete operations using Remote Session Collections.

Operation	REST Call
bulkDelete	POST <server>/services/v2/rest/alarm/bulk/delete

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/RemoteSessionBridge.html>

Create a RemoteSession

```
import com.axeda.services.v2.*
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the alternate ID of an asset that exists and is not muted </li>

 * <li> the name of a remote interface on that asset  </li>
 * <li> a remote server function that is compatible with the {@link
RemoteLaunchOption} type of the interface  </li>
 * </ul>
 */
def assetAlternateId = parameters.assetAlternateId
def interfaceName = parameters.interfaceName
def serverFunction = parameters.serverFunction as RemoteServerFunction


def result = remoteSessionBridge.createSession(new AssetReference(id:
assetAlternateId), interfaceName,
                                               serverFunction, "remote
session description", false, true)
if (result.successful)
{
    println 'session created successfully'
}
else
{
    result.failures.any() {failure ->
        assert failure.code == 'OBJECT_NOT_FOUND'
    }
}
```

Delete a RemoteSession

```
import static com.axeda.drm.sdk.customobject.Call.*
import com.axeda.services.v2.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of a valid remote session </li>
 * </ul>
 */
String remoteSessionId = parameters.remoteSessionId

def result = remoteSessionBridge.deleteHistory(new RemoteSession(systemId:
remoteSessionId))
if (result.successful)
```

```
{
    println 'session deleted successfully'
}
else
{
    result.failures.any() {failure ->
        assert failure.code == 'OBJECT_NOT_FOUND'
    }
}
```

End a RemoteSession

```
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of a valid remote session that has not been previously
ended</li>
 * </ul>
 */

String remoteSessionId = parameters.remoteSessionId

def result = remoteSessionBridge.endSession(remoteSessionId)
if (result.successful)
{
    println 'Session ended successfully'
}
else
{
    assert result.failures.any() { it.code == "ILLEGAL_STATE" }  //already
ended
}
```

Find a RemoteSession by Alternate ID

```
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of a valid remote session</li>
 * </ul>
 */
String remoteSessionAlternateId = parameters.remoteSessionAlternateId


def remoteSession = remoteSessionBridge.find(remoteSessionAlternateId)

assert remoteSession
```

Find a RemoteSession by Criteria

```
import com.axeda.services.v2.*
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the name of an existing asset on which there are active remote
sessions</li>
 * <li> the completed time as null </li>
 * </ul>
 */
String assetName = parameters.assetName
def completedTime = parameters.completedTime as Date

def criteria = new RemoteSessionCriteria()
criteria.setAssetName(assetName)
criteria.setCompletedTime(completedTime)
def result = remoteSessionBridge.find(criteria)

assert !result.getRemoteSessions().isEmpty()
```

Find a RemoteSession by System ID

```
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of a valid remote session</li>
 * </ul>
 */
String remoteSessionId = parameters.remoteSessionId

def remoteSession = remoteSessionBridge.findById(remoteSessionId)
assert remoteSession
```

Find a RemoteSession Interface

```
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the name of an existing interface on an existing asset</li>
 * </ul>
 */
String interfaceId = parameters.interfaceId

def remoteInterface = remoteSessionBridge.findInterface(interfaceId);

```

```
assert remoteInterface
```

Find RemoteSession Interfaces

```
1  import static com.axeda.drm.sdk.customobject.Call.*
2  import com.axeda.services.v2.*
3  import static com.axeda.sdk.v2.dsl.Bridges.*
4
5  /**
6   * For this example to work, you need to pass the following parameters:
7
8   * <ul>
9   * <li> the alternate ID of an existing asset that has remote
10 interfaces</li>
11  * </ul>
12  */
13 def assetAlternateId = parameters.assetAlternateId
14 def remoteInterfaces = remoteSessionBridge.findInterfaces(new
   AssetReference(id: assetAlternateId), true)

   assert !remoteInterfaces.isEmpty()
```

Find One RemoteSession

```
1  import com.axeda.services.v2.*
2  import static com.axeda.drm.sdk.customobject.Call.*
3  import static com.axeda.sdk.v2.dsl.Bridges.*
4
5  /**
6   * For this example to work, you need to pass the following parameters:
7
8   * <ul>
9   * <li> the name of an existing asset on which there are active remote
10 sessions</li>
11  * </ul>
12  */
13 String assetName = parameters.assetName
14
15 def criteria = new RemoteSessionCriteria()
16 criteria.assetName = assetName
17 def session = remoteSessionBridge.findOne(criteria)

   assert session
```

Get RemoteSession Status

```
1  import static com.axeda.drm.sdk.customobject.Call.*
2  import static com.axeda.sdk.v2.dsl.Bridges.*
3
4  /**
5   * For this example to work, you need to pass the following parameters:
6
7   * <ul>
8   * <li>the ID of a valid remote session</li>
9   * </ul>
10  */
11 String remoteSessionId = parameters.remoteSessionId
12
```

```
def status = remoteSessionBridge.getSessionStatus(remoteSessionId)

println 'Session status is: ' + status.value()
```

Validate a RemoteSession User

```
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of a valid remote session</li>
 * <li> the username of an existing user</li>
 * </ul>
 */

String sessionId = parameters.remoteSessionId
def username = parameters.usernameToCheck

def valid = remoteSessionBridge.isUserValidForSession(username,
sessionId);

println String.format("User %s is %s for the session", username, valid ?
"valid" : "not valid")
```

Join a RemoteSession

```
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*


/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of an existing remote session with {@link
RemoteServerFunction#TERMINAL}, of <strong>telnet</strong> type,
 * that should not be terminated or in exclusive mode, on a non muted
asset</li>
 * </ul>
 * The user making the request should have not previously joined the
session.
 */

String remoteSessionId = parameters.remoteSessionId

def result = remoteSessionBridge.joinSession(remoteSessionId)

if (result.successful)
{
    println 'Successfully joined the session'
}

else
{
    assert result.failures.every() { it.code in ["SESSION_NOT_TELNET",
```

```
"SESSION_CREATED_IN_EXCLUSIVE_MODE", "USER_ALREADY_IN_SESSION"] }

}
```

Leave a RemoteSession

```
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of an existing remote session with {@link
RemoteServerFunction#TERMINAL}, of <strong>telnet</strong> type,
 * that should not be terminated</li>
 * </ul>
 * The user making the request should have previously joined the session,
and cannot be the owner of the session.
 */

String remoteSessionId = parameters.remoteSessionId

def result = remoteSessionBridge.leaveSession(remoteSessionId)

if (result.successful)
{
    println 'Successfully left the session'
}

else
{
    assert result.failures.every() { it.code in ["SESSION_NOT_TELNET",
"OWNER_CANT_LEAVE_SESSION", "USER_NOT_IN_SESSION"] }

}
```

Merge RemoteSessions

```
import com.axeda.services.v2.*
import static com.axeda.drm.sdk.customobject.Call.*
import static com.axeda.sdk.v2.dsl.Bridges.*

/**
 *  For this example to work, you need to pass the following parameters:

 * <ul>
 * <li>the name of an existing remote interface on an existing asset </li>

 * <li>a remote server function that is compatible with the {@link
RemoteLaunchOption} type of the interface</li>
 * <li>the ID of an existing remote session created by the current user on
the above remote interface</li>
 * </ul>
 * Neither the existing remote session, nor the new one can be of type
<strong>terminal</strong>.
 */

```

```
String interfaceName = parameters.interfaceName
def serverFunction = parameters.serverFunction as RemoteServerFunction

String existingSessionId = parameters.existingSessionId

def result = remoteSessionBridge.mergeSession(interfaceName,
serverFunction, existingSessionId, "this is a merged session", true)
if (result.successful)
{
  println 'Sessions merged successfully'
}
else
{
  assert result.failures.every() {
    it.code in ["OPERATION_DENIED_BY_CONFIGURATION",
"CANNOT_MERGE_TERMINAL_SESSIONS",
"CANNOT_MERGE_SESSIONS_CREATED_BY_DIFFERENT_USERS"]
  }

}
```

Bulk Delete

```
import com.axeda.drm.sdk.customobject.Call
import com.axeda.services.v2.RemoteSession

import static com.axeda.sdk.v2.dsl.Bridges.getRemoteSessionBridge

/**
 * For this example to work, you need to pass the following parameters:

 * <ul>
 * <li> the ID of a valid remote session </li>
 * <li> the ID of another valid remote session </li>
 * </ul>
 */
String remoteSessionId_1 = Call.parameters.remoteSessionId_1
String remoteSessionId_2 = Call.parameters.remoteSessionId_2

def result = remoteSessionBridge.deleteHistory([new
RemoteSession(systemId: remoteSessionId_1), new RemoteSession(systemId:
remoteSessionId_2)])
assert result.successful
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/RemoteSessionBridge.html>

SoftwarePackageBridge

Software Packages are collections of information used to describe a package that includes instructions intended to be delivered to Assets for evaluation.

Software Packages can be referenced by the internal identifier given to the object by the Axeda Platform, or they can be referenced using a user-generated and unique alternateId. Permissions can be applied to the package, thereby granting different groups access to the package. For example, a Read permission allows the package to be deployed, and a Delete permission allows the package to be deleted.

Until published, a Software Package can be edited. Once published, a Software Package can only be deployed or deleted.

This section provides an overview of creating and manipulating Software Package objects through the Axeda API using the SoftwarePackageBridge. The SoftwarePackageBridge provides Create, Read/Find, Update, and Delete (CRUD) operations for SoftwarePackage objects.

SoftwarePackage Properties

SoftwarePackage objects can contain properties such as name, description, version, model, category, and dependencies. For a full description of SoftwarePackage methods and properties, see the Axeda API Specification.

Dependencies

Software Packages can have Dependencies. Dependencies are a list of DependencyExpressions. A DependencyExpression consists of a type, a target, an operator and a value.

Dependency Types:

- Package — limits the package to Assets that have had another package deployed to them.
- Data Item — limits the package to Assets that have a data item that satisfies a given condition.
- Registry — limits the package to Assets that have a certain value in their dependencies.xml file for a parameter you want to test.
- Defined Configuration — limits the package to Assets that match the defined configuration.

Dependency Type	Formula	Example
Package	${Dependent Package Name}	>= Software Update||1.2.4.5 (Package Name || version)
Data Item	${Data Item Name} ${Operator}	Temperature >= 100
Registry	${Registry Item Name} ${Operator} ${Value}	version <= 2
Defined	(any operators will be simply	The Defined Configuration == My Defined

Required Properties

- name
- a valid, existing model
- version
- at least one instruction

Uniqueness constraints

Every SoftwarePackage must have a unique name and version.

Default values

primaryAgentsOnly - “true”

Expiration Date

Software Packages can be expired by setting the expirationDate. At the specified time, the package will be no longer be available for deployments.

Methods

The following table provides a brief overview of the available SoftwarePackageBridge methods. For a full description of SoftwarePackageBridge methods, see the Axeda API Specification.

Method	Description
create	Creates a new Software Package.
delete	Deletes a Software Package.
find (by criteria)	Finds Software Packages based on search criteria.
find (by IDs)	Finds Software Packages based on a list of identifiers.
find (by alternate ID)	Finds a Software Package based on the alternate identifier.
findById	Finds a Software Package based on its platform identifier.
findOne	Returns the first Software Package found that meets specified search criteria.
generateAlternateId	Returns a unique alternate identifier for the specified Software Package.
publish	Publishes the specified Software Package, which locks it from Update.
reset (by ID)	Resets Software Packages based on the specified assetId. Will not reset packages that are in the IN_PROGRESS or CREATED state.
reset (by criteria)	Resets Software Packages based on the specified criteria. Will not reset packages that are in the IN_PROGRESS or CREATED state.
toString	Generates a string representing a specified Software Package.
update	Updates an existing Software Package.

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/SoftwarePackageBridge.html>

Create a SoftwarePackage

```
1  import com.axeda.services.v2.SoftwarePackage
2  import com.axeda.sdk.v2.dsl.Bridges
3  import com.axeda.services.v2.SetDataItemInstruction
4  import com.axeda.services.v2.NamedValue
5
6  //Create software package
7  def sPkg = new SoftwarePackage();
8  sPkg.name = "hello_test_package"
9  sPkg.version = "1.2.4.7"
10 sPkg.model = find.model(parameters.modelNumber)
11
12 //Create a instruction (Execute Application)
13 def setDIInstruction = new SetDataItemInstruction()
14 def namedValue = new NamedValue()
15 namedValue.name = "hello_dataItem"
16 namedValue.value = "100"
17 setDIInstruction.dataItemValue = namedValue
18
19 sPkg.getInstructions().add(setDIInstruction)
20
21 return Bridges.softwarePackageBridge.create(sPkg)
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/SoftwarePackageBridge.html>

UserBridge

In order to interact with the Axeda Patform, a user must be defined in a User object. The User object contains information such as user name, password, and contact information. You can use the User object apiOnly property to restrict a user from accessing the Axeda Console UI. Setting apiOnly also prevents the user's password from expiring.

Access to Platform objects can be granted or restricted with User Groups. A User assigned to a User Group inherits all the access privileges associated with that User Group. For Users not assigned to the User Group, access is restricted.

Users can be created using the Axeda Console, through the Axeda API, or via a REST call over HTTP. This section provides an overview of creating and manipulating Users through the API using the UserBridge object.

The UserBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for User objects. You can also use the UserBridge to expire user passwords, get password validation policies, unlock Users, and to issue requests to reset user passwords.

Methods

The following table provides a brief overview of the UserBridge methods, along with available REST calls. For a full description of UserBridge methods, see the Axeda API Specification.

Method	Description	REST Call
create	Creates a new User.	PUT <server>/services/v2/rest/user **Note:** this same REST call as a POST invokes a Save operation: POST <server>/services/v2/rest/user
delete	Deletes a User.	DELETE <server>/services/v2/rest/user/id/{i
expirePassword	Sets the status of the referenced User's password to Expired.	PUT <server>/services/v2/rest/user/password/expire
find (by criteria)	Finds Users based on search criteria.	POST
find (by IDs)	Finds Users based on a list of identifiers.	POST <server>/services/v2/rest/user/find
find (by alternate ID)	Finds a User based on the alternate identifier.	GET
findById	Finds a User based on its platform identifier.	GET
findOne	Returns the first User found that meets specified search criteria.	POST <server>/services/v2/rest/user/find
getPasswordValida	Returns information about the current	GET

tionPolicy	password validation policy configuration.	<server>/services/v2/rest/user/validationPolicy
resetPasswordRequest	Issues a request to reset the password of the specified user.	POST <server>/services/v2/rest/user/password/resetRequest
toString	Generates a string representing a specified	N/A
unlockUser	Unlocks the referenced locked out User.	PUT
update	Updates an existing User.	POST <server>/services/v2/rest/user/id/{i

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations using User Collections.

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/user/bulk/create
bulkDelete	POST <server>/services/v2/rest/user/bulk/delete
bulkSave	POST <server>/services/v2/rest/user/bulk/save
bulkUpdate	POST <server>/services/v2/rest/user/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/UserBridge.html>

Create a User

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def user = new User(
    username: "MikeDoe",
    password: "Encrypted1!",
    fullName: "Mike Doe",
    emailAddresses: ["mike.doe@axeda.com"],
    locale: new Locale(countryCode: java.util.Locale.US.getCountry(),
languageCode: java.util.Locale.US.getLanguage())
)

def result = userBridge.create(user)
if (result.successful) {
  println "User created successfully"
}
else {
  assert result.failures.any() { it.code == "OBJECT_ALREADY_EXISTS" }
}
```

Delete a User

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def user = new User(id: "StevenDoe")

def result = userBridge.delete(user)
if (result.successful) {
  println "The user was deleted"
}
else {
  assert result.failures.every { it.code == "OBJECT_NOT_FOUND" }
}
```

Change a User's Expired Password

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def result = userBridge.changeExpiredPassword(new UserReference(id:
'JoeBloggs'), 'Encrypted1!', 'Pass1234_')

if (result.successful) {
  println "User's expired password was successfully changed"
}
else {
  assert result.failures.any() { it.code == "AUTHENTICATION_FAILURE" ||
it.code == "UNSUPPORTED_OPERATION" }
}
```

Change a User's Password

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def result = userBridge.changePassword(new UserReference(id:

if (result.successful) {
  println 'User password was successfully expired'
}
else {
  assert result.failures.any() { it.code == "AUTHENTICATION_FAILURE" ||
it.code == "INVALID_PASSWORD" }
}
```

Expire a User's Password

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def result = userBridge.expirePassword(new UserReference(id:

if (result.successful) {
  println 'User password was successfully expired'
}
else {
  assert result.failures.any() { it.code == "PERMISSION_DENIED" }
}
```

Find a User by Alternate ID

```
import static com.axeda.sdk.v2.dsl.Bridges.*

def user = userBridge.find("JohnDoe")
assert user;
```

Find a User by Criteria

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def findResult = userBridge.find(new UserCriteria(timezoneId: 'PST'))
assert !findResult.users.isEmpty()
```

Find a User by System ID

```
import com.axeda.drm.sdk.customobject.*
import static com.axeda.sdk.v2.dsl.Bridges.*

String userId = Call.parameters.userId

def foundUser = userBridge.findById(userId)
assert foundUser
```

Find One User

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def findResult = userBridge.findOne(new UserCriteria(username: 'J*'))
assert findResult
```

Reset Password Request

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def result = userBridge.resetPasswordRequest(new UserReference(id:
'JoeBloggs'), "server")

if (result.successful) {
  println 'Request completed successfully'
}
else {
  assert result.failures.any() { it.code == "ILLEGAL_STATE" }
}
```

Save a User

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def user = new User(
    username: "NikiDoe",
    password: "Encrypted1!",
    fullName: "Niki Doe",
    locale: new Locale(languageCode: java.util.Locale.US.getLanguage())
)

def result = userBridge.save(user)
if (result.successful) {
  println "User saved successfully"
}
else {
  assert result.failures.any() { it.code == "OBJECT_ALREADY_EXISTS" }
}
```

Unlock a User

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def result = userBridge.unlockUser(new UserReference(id: 'JohnStiles'))

if (result.successful) {
  println 'User was successfully unlocked'
}
else {
  assert result.failures.any() { it.code == "PERMISSION_ADMIN" }
}
```

Update a User

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def user = new User(
    id: "JohnDoe",
    username: "JohnDoe",
    fullName: "John Doe",
    timezoneId: "PST",
    locale: new Locale(countryCode: java.util.Locale.US.getCountry(),
languageCode: java.util.Locale.US.getLanguage())
)

def result = userBridge.update(user)
if (result.successful) {
  println "User updated successfully"
}
else {
  assert result.failures.any() { it.code == "OBJECT_NOT_FOUND" }
```

Validate Password (new password)

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def result = userBridge.validatePassword(new UserReference(id:
'Richard_Miles'), 'Pass1234_')

if (result.successful) {
  println 'The new password would be valid'
}
else {
  assert result.failures.every() { it.code == "INVALID_PASSWORD" }
}
```

Validate Password (current and new password)

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def result = userBridge.validatePassword(new UserReference(id:
'Richard_Miles'), 'Encrypted1!', 'Encrypted1#')

if (result.successful) {
  println 'The password is valid'
}
else {
  assert result.failures.every() { it.code == "INVALID_PASSWORD" }
}
```

Bulk Create

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def user1 = new User(
    username: "SmithDoe",
```

```
    password: "Encrypted1!",
    fullName: "Smith Doe",
    emailAddresses: ["smith.doe@axeda.com"],
    locale: new Locale(languageCode: java.util.Locale.US.getLanguage())
)

def user2 = new User(
    username: "JoanaDoe",
    password: "Encrypted1!",
    fullName: "Joana Doe",
    timezoneId: "PST",
    locale: new Locale(countryCode: java.util.Locale.US.getCountry(),
languageCode: java.util.Locale.US.getLanguage())
)

def result = userBridge.create([user1, user2])
if (result.successful) {
  println "Users created successfully"
}
else {
  assert result.failures.every() { it.code == "OBJECT_ALREADY_EXISTS" }
}
```

Bulk Delete

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*


def result = userBridge.delete([new User(id: "JamesSmith"), new User(id:
if (result.successful) {
  println "The user was deleted"
}
else {
  assert result.failures.every { it.code == "OBJECT_NOT_FOUND" }
}
```

Bulk Save

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

/*  create this user */
def user1 = new User(
    username: "AlanDoe",
    password: "Encrypted1!",
    fullName: "Alan Doe",
    emailAddresses: ["alan.doe@axeda.com"],
    locale: new Locale(countryCode: java.util.Locale.US.getCountry(),
languageCode: java.util.Locale.US.getLanguage())
)

/*  update this user */
def user2 = new User(
    id: "JaneDoe",
    username: "JaneDoe",
    fullName: "Jane Doe updated",
    timezoneId: "PST",
    locale: new Locale(languageCode: java.util.Locale.US.getLanguage())
```

```
)

def result = userBridge.save([user1, user2])
if (result.successful) {
  println "Users saved successfully"
}
else {
  assert result.failures.every() { it.code == "OBJECT_ALREADY_EXISTS" }
}
```

Bulk Update

```
import static com.axeda.sdk.v2.dsl.Bridges.*
import com.axeda.services.v2.*

def user1 = new User(
    id: "JohnDoe",
    username: "JohnDoe",
    fullName: "John Doe",
    timezoneId: "PST",
    locale: new Locale(countryCode: java.util.Locale.US.getCountry(),
languageCode: java.util.Locale.US.getLanguage())
)

def user2 = new User(
    id: "JaneDoe",
    username: "JaneDoe",
    fullName: "Jane Doe",
    emailAddresses: ["JaneDoe2@yahoo.com"],
    locale: new Locale(countryCode: java.util.Locale.UK.getCountry(),
languageCode: java.util.Locale.UK.getLanguage())
)

def result = userBridge.update([user1, user2])
if (result.successful) {
  println "Users updated successfully"
}
else {
  assert result.failures.any() { it.code == "OBJECT_NOT_FOUND" }
}
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/UserBridge.html>

Create a User

Endpoint	/services/v2/rest/user
Verb(s)	PUT
Input Object	User
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<User xmlns="http://www.axeda.com/services/v2">
  <username>user_1367591520222</username>
  <fullName>full_1367591520222</fullName>
  <password>pass4tests</password>
  <emailAddresses>user_1367591520222@host.com</emailAddresses>
  <locale>
    <languageCode>en</languageCode>
    <countryCode>US</countryCode>
  </locale>
</User>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded>
    <v2:success>
      <v2:ref>user_1367591520222</v2:ref>
      <v2:id>50</v2:id>
    </v2:success>
  </v2:succeeded>
    <v2:failures/>
</v2:ExecutionResult>
```

Save a User

Endpoint	/services/v2/rest/user
Verb(s)	POST
Input Object	User
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<User xmlns="http://www.axeda.com/services/v2">
  <username>user_1367591521959</username>
  <fullName>full_1367591521959</fullName>
  <password>pass4tests</password>
  <emailAddresses>user_1367591521959@host.com</emailAddresses>
  <locale>
    <languageCode>en</languageCode>
```

```
    <countryCode>US</countryCode>
  </locale>
</User>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded>
    <v2:success>
      <v2:ref>user_1367591521959</v2:ref>
      <v2:id>51</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Delete a User

Endpoint	/services/v2/rest/user/id/52
Verb(s)	DELETE
Input Object	
Output Object	ExecutionResult

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded>
    <v2:success>
      <v2:ref>52</v2:ref>
      <v2:id>52</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Update a User

Endpoint	/services/v2/rest/user/id/49
Verb(s)	POST
Input Object	User
Output Object	ExecutionResult

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<User xmlns="http://www.axeda.com/services/v2" systemId="49">
  <username>user_1367591518488</username>
  <fullName>full_1367591518488</fullName>
  <emailAddresses>user@host.com</emailAddresses>
  <mailingAddress>new mailing addr</mailingAddress>
  <locale>
    <languageCode>it</languageCode>
```

```
  </locale>
</User>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:ExecutionResult xmlns:v2
="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <v2:succeeded>
    <v2:success>
      <v2:ref>49</v2:ref>
      <v2:id>49</v2:id>
    </v2:success>
  </v2:succeeded>
  <v2:failures/>
</v2:ExecutionResult>
```

Find a User by Criteria

Endpoint	/services/v2/rest/user/find
Verb(s)	POST
Input Object	UserCriteria
Output Object	FindUserResult

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<UserCriteria xmlns="http://www.axeda.com/services/v2"
  <username>*1367591508653*</username>
</UserCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:FindUserResult xmlns:v2
="http://www.axeda.com/services/v2"
  <v2:criteria pageSize="10">
    <v2:username>*1367591508653*</v2:username>
  </v2:criteria>
  <v2:users>
    <v2:users xsi:type="v2:User" id="user_13675915086531"
systemId="39" label="full_13675915086531" detail="user_
13675915086531@host.com" restUrl="http://jtelarico-
      <v2:username>user_13675915086531</v2:username>
      <v2:fullName>full_13675915086531</v2:fullName>
      <v2:phoneNumber/>
      <v2:faxNumber/>
      <v2:emailAddresses>user_13675915086531
      <v2:mailingAddress/>
      <v2:registeredAddress/>
      <v2:city/>
      <v2:state/>
      <v2:zipCode/>
      <v2:locale>
        <v2:languageCode>en</v2:languageCode>
        <v2:countryCode>US</v2:countryCode>
      </v2:locale>
      <v2:timezoneId>GMT</v2:timezoneId>
      <v2:active>true</v2:active>
      <v2:admin>false</v2:admin>
      <v2:apiOnly>false</v2:apiOnly>
      <v2:userStatus>
```

```
        <v2:passwordExpired>false</v2:passwordExpired>
        <v2:passwordExpiresOn>2013-08-01T10:31:49.534-04:00<
/v2:passwordExpiresOn>
        <v2:locked>false</v2:locked>
      </v2:userStatus>
    </v2:users>
    <v2:users xsi:type="v2:User" id="user_13675915086532"
systemId="40" label="full_13675915086532" detail="user_
13675915086532@host.com" restUrl="http://jtelarico-
      <v2:username>user_13675915086532</v2:username>
      <v2:fullName>full_13675915086532</v2:fullName>
      <v2:phoneNumber/>
      <v2:faxNumber/>
      <v2:emailAddresses>user_13675915086532
      <v2:mailingAddress/>
      <v2:registeredAddress/>
      <v2:city/>
      <v2:state/>
      <v2:zipCode/>
      <v2:locale>
        <v2:languageCode>en</v2:languageCode>
        <v2:countryCode>US</v2:countryCode>
      </v2:locale>
      <v2:timezoneId>PST</v2:timezoneId>
      <v2:active>true</v2:active>
      <v2:admin>false</v2:admin>
      <v2:apiOnly>false</v2:apiOnly>
      <v2:userStatus>
        <v2:passwordExpired>false</v2:passwordExpired>
        <v2:passwordExpiresOn>2013-08-01T10:31:50.532-04:00<
/v2:passwordExpiresOn>
        <v2:locked>false</v2:locked>
      </v2:userStatus>
    </v2:users>
  </v2:users>
</v2:FindUserResult>
```

Find One User

Endpoint	/services/v2/rest/user/findOne
Verb(s)	POST
Input Object	UserCriteria
Output Object	User

XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<UserCriteria xmlns="http://www.axeda.com/services/v2">
  <username>user_user11367591526499</username>
</UserCriteria>
```

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<v2:User xmlns:v2="http://www.axeda.com/services/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="user_user11367591526499" systemId="54"
label="full_user11367591526499"
detail="user_user11367591526499@host.com"
  <v2:username>user_user11367591526499</v2:username>
```

```
  <v2:fullName>full_user11367591526499</v2:fullName>
  <v2:phoneNumber/>
  <v2:faxNumber/>
  <v2:emailAddresses>user_user11367591526499
  <v2:mailingAddress/>
  <v2:registeredAddress/>
  <v2:city/>
  <v2:state/>
  <v2:zipCode/>
  <v2:locale>
    <v2:languageCode>en</v2:languageCode>
    <v2:countryCode>US</v2:countryCode>
  </v2:locale>
  <v2:timezoneId>GMT</v2:timezoneId>
  <v2:active>true</v2:active>
  <v2:admin>false</v2:admin>
  <v2:apiOnly>false</v2:apiOnly>
  <v2:userStatus>
    <v2:passwordExpired>false</v2:passwordExpired>
    <v2:passwordExpiresOn>2013-08-01T10:32:07.195-04:00</v2:pa
sswordExpiresOn>
    <v2:locked>false</v2:locked>
  </v2:userStatus>
</v2:User>
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/UserBridge.html>

UserSettingsBridge

The UserSettingsBridge object provides Create, Read/Find, Update, and Delete (CRUD) operations for the Platform Settings User layer.

The configuration settings in the platform are structured in layers. The important idea is that each layer will be overlayed on the previous layer to form a complete view of the settings. Any setting defined in a lower layer will overwrite the same setting defined in an upper layer, if present. The three layers defined in the platform are (from top to bottom, with the Default layer being the highest layer, and the User layer being the lowest layer):

The Default layer — This layer contains the default settings specified by Axeda, and these settings can have any path. These configurations can be read by the install layer and they can be overwritten by the install and user layers.

The Installation layer — This layer contains any settings set by the installation administrator. This layer can contain settings with any path under /settings. These settings will overwrite the settings already defined in the Default layer. The InstallSettingsBridge is used to manage the settings in this layer.

The User layer — This layer contains personalized settings for each user. Access to this layer is based on the currently logged in user. The User layer cannot read or write to /settings/system path, as this path is reserved for the Default and Installation layers. Except for this reserved path, the User layer can contain settings with any other path under /settings. These settings will overwrite the settings defined in the Installation and Default layers. The UserSettingsBridge is used to manage the settings in this layer.

Methods

The UserSettingsBridge is used to manage the settings in the User layer.

The following table provides a brief overview of the UserSettingsBridge methods, along with available REST calls. For a full description of UserSettingsBridge methods, see the Axeda API Specification.

Method	Description	REST Call
create	Creates a new PlatformSettings object.	PUT <server>/services/v2/rest/settings/user
delete	Deletes a PlatformSettings object.	DELETE <server>/services/v2/rest/settings/user
find (String path)	Finds the PlatformSettings object containing all the settings that match the specified path.	GET <server>/services/v2/rest/settings/user
findById	Since the PlatformSettings object does not have a system id, and it	GET

	can solely be identified by the root path, this method does the same	<server>/services/v2/rest/settings/user
fromXML	Builds a PlatformSettings object from the specified XML string.	N/A
generateAlternateId	A PlatformSettings object can be identified by its rootPath.	N/A
insertSetting	Updates the specified PlatformSettings instance by inserting a setting with the specified path and value.	N/A
removeNode	Updates the PlatformSettings by removing all the settings matching the specified nodePath.	N/A
removeSetting	Removes the specified Setting from the specified PlatformSettings	N/A
toString	Generates a string representing a specified Expression Rule.	N/A
toXML	Generates a XML string representing the PlatformSettings object.	N/A
update	Updates/replaces all the existing settings under the specified PlatformSettings.rootPath.	PUT <server>/services/v2/rest/settings/user
updateSettin	Updates the value of the setting that belongs to the specified	N/A

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/UserSettingsBridge.html>

Wednesday, March 01, 2017 10:40 AM

WebResourceBridge

The Web Resource API provides a way for developers to automatically generate client classes from web service WSDLs specified by remote URLs. The Axeda Platform creates all Web Service method calls provided by that WSDL. Developers can implement custom objects that use the resulting methods.

This functionality operates with SOAP-based, v1.1 and v1.2 WSDLs only. This functionality generates a Maven artifact. Knowledge of Maven makes project creation and development much easier. If you choose to use the Maven artifact, you need an installation of Apache Maven, v2.x or later. For more information or to download Maven, see http://maven.apache.org/download.html.

Workflow

The expected workflow for converting Web Resources and using the resulting classes is as follows:

1. Create the Web resource client class by calling the create operation with the WSDL to convert. (You can convert one or more Web Resouces in one operation.)

During Web Resource creation, each WSDL specified in the resource is downloaded. To download a secure WSDL, only HTTP Basic authentication is provided. Each WSDL is converted into client classes, then added to a Maven-like JAR artifact. The URLs are returned in the new Web Resource domain object.

The Jar is available for customers to download. The Maven Dependency section provides an XML Maven dependency definition and repository definition that can be used by developers for integration in their IDEs (for example, IntelliJ IDEA, Community Editions).

2. Import the client library created from the Web resource conversion into your project. To do this, you can do one of the following:

Download the JAR directly from the download path specified in the returned Web resource domain object and then manually import the JAR into your IDE. For example, the XML for this download path would look similar to:

<v2:jarDownloadURL>
http://<server_name:port>/services/v1/rest/webresource/repo/com/axeda/webresource/1/weather/1.0.0/weather-1.0.0.jar</v2:jarDownloadURL

-or-

Update your Maven POM file for the new artifact and download it from your Axeda Platform. This method simplifes the download/integration process.

3. Create Groovy scripts in your IDE, using the endpoints provided in the new Web resource client class.

4. Upload your scripts to the Axeda Platform and configure them to be run as Custom Objects.

Using the Maven Repository Artifact

When you create a new Web Resource, the Axeda Platform generates a Maven artifact with the client classes for the Web resource.

The domain object provides a Maven Dependency section with an XML Maven artifact dependency definition and repository definition. This information can be used for integration in your IDE.

The following is an example of a Maven Dependency XML definition:

```
<dependency>
 <groupId>com.axeda.webresource.1</groupId>
 <artifactId>weather</artifactId>
 <version>1.0.0</version>
</dependency>
```

```
<repository>
 <id>axeda.webresource</id>
 <url>http://<server>/services/v1/rest/webresource/repo</url>
</repository>
```

where: *<server>* is replaced with the Axeda Platform URL

You can copy and paste this XMLinto your Maven pom.xml and download the Web Resource JAR from the Axeda Platform.

Endpoints Defined in Web Resource Object

Any endpoint classes available in the WSDL are available for consumption in your IDE. For each endpoint, the Web resource provides its location and sample source code that you can copy and paste into your IDE to get started.

The following example Web Resouce, *Weather*, provides the endpoint, *WeatherHttpGet*, at *com.axeda.webresource.weather.WeatherHttpGet*: The source code snippet (`<v2:exampleSourceCode>`) shows how to call the related endpoint. You can copy and paste it into your IDE as a starting point.

```
<repository>
 <id>axeda.webresource</id>
 <url>http://rbutnaru-lt7.axeda.com:8080/services/v1/rest/webresource/repo</url>
</repository>]]></v2:mavenDependencyXML>
 <v2:services>
   <v2:service id="506" label="Weather" detail="com.axeda.webresource.weather.Weather">
     <v2:name>Weather</v2:name>
     <v2:className>com.axeda.webresource.weather.Weather</v2:className>
```

```
    <v2:endpoints>
        <v2:endpoint id="522" label="WeatherHttpGet"
detail="com.axeda.webresource.weather.WeatherHttpGet">
            <v2:name>WeatherHttpGet</v2:name>
            <v2:interfaceClass>com.axeda.webresource.weather.WeatherHttpGet</v2:interfaceClass>
            <v2:defaultEndpointURL>
http://wsf.cdyne.com/WeatherWS/Weather.asmx</v2:defaultEndpointURL>
            <v2:exampleSourceCode>import com.axeda.sdk.v2.bridge.WebResourceBridge
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.webresource.weather.WeatherHttpGet

WebResourceBridge bridge = Bridges.webResourceBridge
WeatherHttpGet endpoint = bridge.getClientEndpoint("Weather Service", "Weather", "WeatherHttpGet")
</v2:exampleSourceCode>
        </v2:endpoint>
```

Methods

The following table provides a brief overview of the available WebResourceBridge methods, along with available REST calls. For a full description of WebResourceBridge methods, see the Axeda API Specification.

Method	Description	Rest Call
addHttpHeader	Adds the HTTP header to the specified client endpoint using the specified field and value.	N/A
addHttpHeader (list)	Adds the HTTP header to the specified client endpoint using the specified field and list of values.	N/A
create	Creates a new WebResource object.	PUT <server>/services/v2/rest/webResource **Note:** this same REST call as a POST invokes a Save POST <server>/services/v2/rest/webResource
delete	Deletes a WebResource object.	DELETE <server>/services/v2/rest/WebResource/id/{id}
find (by criteria)	Finds WebResource objects based on search criteria.	POST <server>/services/v2/rest/webResource/find
find (by IDs)	Finds WebResource objects based on a list of	POST <server>/services/v2/rest/webResource/findByIds

find (by alternate ID)	Finds a WebResource object based on the alternate	GET <server>/services/v2/rest/webResource/id/{id}
findById	Finds a WebResource object based on its platform	GET <server>/services/v2/rest/webResource/id/{id}
findOne	Returns the first WebResource object found that meets specified search criteria.	POST <server>/services/v2/rest/webResource/findOne
getClientEndpoint	Creates a web services client endpoint object for the specified WebResource name, Service name, and	N/A
setEndpointURL	Sets the endpoint URL for the specified client	N/A
setHttpBasicAuthCredentials	Sets HTTP basic authentication credentials for the specified client endpoint.	N/A
setWSSUsernameTokenCredentials	Sets WSS username token authentication credentials for the specified client endpoint.	N/A
toString	Generates a string representing a specified	N/A
update	Updates an existing WebResource object.	POST <server>/services/v2/rest/webResource/id/{id}

Bulk REST Calls

In the REST style, you can send in a collection of the identified resource for all Create, Update, Save, and Delete operations. The Find verb will return either a single instance or a collection, depending on how many results were found.

The following REST Calls can be used for bulk operations:

Operation	REST Call
bulkCreate	POST <server>/services/v2/rest/webResource/bulk/create
bulkDelete	POST <server>/services/v2/rest/webResource/bulk/delete
bulkSave	POST <server>/services/v2/rest/webResource/bulk/save
bulkUpdate	POST <server>/services/v2/rest/webResource/bulk/update

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/WebResourceBridge.html>

Create a WebResource

```
import com.axeda.sdk.v2.bridge.WebResourceBridge
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.services.v2.WebResource

def name = parameters.name
def description = parameters.description
def packageNamespace = parameters.packageNamespace
def wsdlUrls = parameters.wsdlUrls
def wsdlUsername = parameters.wsdlUsername
def wsdlPassword = parameters.wsdlPassword

WebResource webResource = new WebResource()
webResource.name = name
webResource.description = description
webResource.packageNamespace = packageNamespace
if(wsdlUrls) {
  wsdlUrls.split(";").each() {
  wsdlUrl -> webResource.wsdlURLs.add(wsdlUrl)
  }
}
webResource.wsdlUsername = wsdlUsername
webResource.wsdlPassword = wsdlPassword

WebResourceBridge webResourceBridge = Bridges.webResourceBridge

// persist web resource
def result = webResourceBridge.create(webResource)
if(result.succeeded){
  // return the WebResource object serialized to ZML
  return webResourceBridge.find(result.succeeded.get(0).id)
}
else
{
  // if the create failed, just return the result since we want to know
why it failed
  return result;
}
```

Update a WebResource

```
package WebResource

import com.axeda.sdk.v2.bridge.WebResourceBridge
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.services.v2.WebResource
import com.axeda.services.v2.WebResourceCriteria
import com.axeda.services.v2.ExecutionResult

// import the parameters
def name = parameters.name
def newName = parameters.newName
```

```
def newDescription = parameters.newDescription

WebResourceBridge webResourceBridge = Bridges.webResourceBridge

WebResourceCriteria criteria = new WebResourceCriteria()
criteria.name = name
WebResource webResource = webResourceBridge.findOne(criteria);
if(webResource)
{
  if(newName == null && newDescription == null)
  {
    return "ERROR: Nothing to update since newName and newDescription
parameters are both null"
  }

  if(newName)
  {
    webResource.name = newName
  }

  if(newDescription)
  {
    webResource.description = newDescription
  }
  ExecutionResult result = webResourceBridge.update(webResource)
  if(result.succeeded)
  {
    return webResourceBridge.find(result.succeeded.get(0).id)
  }
  else
  {
    return result;
  }
}
else
{
  return "Cannot find Web Resource for name " + name
}
```

Delete a WebResource

```
package WebResource

import com.axeda.sdk.v2.bridge.WebResourceBridge
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.services.v2.WebResource

// import the parameters
def name = parameters.name

WebResourceBridge webResourceBridge = Bridges.webResourceBridge

// find Web Resource by name
WebResource resource = webResourceBridge.find(name);
if(resource != null)
{
  return webResourceBridge.delete(resource)
}
else
{
```

```
  return "Cannot find Web Resource for name " + name
}
```

Find a WebResource by Criteria

```
package WebResource

import com.axeda.sdk.v2.bridge.WebResourceBridge
import com.axeda.sdk.v2.dsl.Bridges
import com.axeda.services.v2.WebResourceCriteria

// import the parameters
def id = parameters.id
def description = parameters.description
def jar = parameters.jar
def jarDownloadUrl = parameters.jarDownloadUrl
def name = parameters.name
def packageNamespace = parameters.packageNamespace
def wsdlUrl = parameters.wsdlUrl
def sortAscending = parameters.sortAscending

WebResourceBridge webResourceBridge = Bridges.webResourceBridge
WebResourceCriteria criteria = new WebResourceCriteria()

// if id is specified, then all other critiera fields are ignored
if(id){
  criteria.id = id
}
else {
  criteria.description = description
  criteria.jar = jar
  criteria.jarDownloadURL = jarDownloadUrl
  criteria.name = name
  criteria.packageNamespace = packageNamespace
  criteria.wsdlURL = wsdlUrl
}

// sortAscending is set to true by default
if(sortAscending == "false"){
  criteria.sortAscending = false
}

return webResourceBridge.find(criteria);
```

From <https://mentor.axeda.com/magnoliaPublic/mentor/objects/WebResourceBridge.html>