RSA ARCHER® SUITE

RESTful API Reference Guide

6.3

Contact Information

Go to the RSA corporate web site for regional Customer Support telephone and fax numbers: https://community.rsa.com/community/rsa-customer-support.

Trademarks

RSA, the RSA Logo, RSA Archer, RSA Archer Logo, and Dell are either registered trademarks or trademarks of Dell Corporation ("Dell") in the United States and/or other countries. All other trademarks used herein are the property of their respective owners. For a list of RSA trademarks, go to www.emc.com/legal/emc-corporation-trademarks.htm.

License agreement

This software and the associated documentation are proprietary and confidential to Dell, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability.

This software is subject to change without notice and should not be construed as a commitment by Dell.

Third-party licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed on RSA.com. By using this product, a user of this product agrees to be fully bound by terms of the license agreements.

Note on encryption technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

For secure sites, Dell recommends that the software be installed onto encrypted storage for secure operations.

For customers in high security zones, Dell recommends that a full application sanitization and reinstallation from backup occur when sensitive or classified information is spilled.

Note on Section 508 Compliance

The RSA Archer® Suite is built on web technologies which can be used with assistive technologies, such as screen readers, magnifiers, and contrast tools. While these tools are not yet fully supported, RSA is committed to improving the experience of users of these technologies as part of our ongoing product road map for RSA Archer.

The RSA Archer Mobile App can be used with assistive technologies built into iOS. While there remain some gaps in support, RSA is committed to improving the experience of users of these technologies as part of our ongoing product road map for the RSA Archer Mobile App.

Distribution

Use, copying, and distribution of any Dell software described in this publication requires an applicable software license.

Dell believes the information in this publication is accurate as of its publication date. The information is subject to change without notice. Use of the software described herein does not ensure compliance with any laws, rules, or regulations, including privacy laws that apply to RSA's customer's businesses. Use of this software should not be a substitute for consultation with professional advisors, including legal advisors. No contractual obligations are formed by publication of these documents.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." DELL INC. MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

October 2017

Contents

Chapter 1: RSA Archer RESTful API

The RSA Archer® Suite RESTful service is a collection of resources organized in functional segments that are accessed through controllers. Each resource can be acted upon either individually, by key/ID, or in batch. Collections are used broadly to submit requests and construct responses.

RESTful stands for Representational State Transfer. RESTful services are a simpler alternative to SOAP services. Like SOAP services, RESTful services enable a user to access the platform data.

A RESTful service meets all of the following criteria:

- Identification of resources
- Manipulation of resources through representations
- Self-descriptive messages
- Hypermedia as the engine of application state

A RESTful service meets some but not all of the criteria.

The RESTful API uses the JavaScript Object Notation (JSON) format by default for requests and responses, but also supports XML. After a resource is identified, Create, Read, and Delete operations can be performed against the resource using standard HTTP verbs to indicate which action should be used.

The response examples in this guide are limited, as responses vary based on the originating requests.

Sample: JSON response

The response results will vary based on the request. Some responses may be very complex based on the originating call, for example, Get All Applications.

The following example shows the JSON response for a request for a list of applications by Name and Description only:

```
[{
"Links": [],
"RequestedObject": {
"Name": "Technologies",
"KeepLicensed":false,
"IsDeprecated":false,
"Description": "<html><head><style type=\"text/css\">.c0 { font-
family: 'Arial' } .c1 { margin: 0px 0px 13px } </style></head><p
class=\"c1\">The Technologies application provides a searchable
and extensible repository of technology version information that
can be leveraged to relate objects of like technology.</p></html>"
},
```

```
"IsSuccessful": true,
"ValidationMessages": []
},
{
"Links": [],
"RequestedObject": {
"Name": "Test Application 10",
"Description": null
},
"IsSuccessful": true,
"ValidationMessages": []
},
{
"Links": [],
"RequestedObject": {
"Name": "Test Application",
"Description": "This is my description"
},
"IsSuccessful": true,
"ValidationMessages": []
},
{
"Links": [],
"RequestedObject": {
"Name": "Test Application 2",
"Description": "This is my description"
},
"IsSuccessful": true,
"ValidationMessages": []
},
{
"Links": [],
"RequestedObject": {
"Name": "Test Application 3",
"Description": "This is my description"
},
"IsSuccessful": true,
"ValidationMessages": []
},
{
"Links": [],
"RequestedObject": {
"Name": "Mobile Packaging Application",
"Description": "This is my description"
```

```
},
"IsSuccessful": true,
"ValidationMessages": []
}]
```

RESTful service criteria

The following table describes how RSA Archer has implemented RESTful services for the mobile app.

Criteria	Implementation
Identification of resources	Instead of exposing methods that can be called, RSA Archer exposes resources that can be retrieved. For example, instead of listing methods that have names with verbs in them, such as UpdateQuestionnaire. RSA Archer exposes questionnaires as resources with the means to interact with them using standard HTTP verbs like GET and POST.
Manipulation of resources through representations	Rather than having a service return strongly typed objects (C# objects with all the expected properties), RSA Archer returns representations of those resources. The caller determines the form that these representations take. • If the caller wants a response in JSON, the Accept header must contain application/json. • If the caller wants a response in XML, the Accept header must contain application/xml.
Self-descriptive messages	Rather than relying on the method name having meaning for RSA Archer to figure out how the method interacts with the resource, HTTP verbs are used. Each message describes itself. • If the resource is accessed at /api/core/application and the method of the request is POST (POST to /api/core/application/vendors), the user is trying to save the new application called Vendors. • If the request is a GET to /api/core/application/vendors, the user is requesting application vendors.

Criteria	Implementation
Hypermedia as the engine of application state	Instead of knowing the API that interacts with RPC methods, a RESTful API provides the root URI and a place to start for interacting with the resource. Each response contains the links that help move through the rest of the API. For example, when the user sends a GET to /api/core/application/vendors, the response includes a link that shows how to update that application (PUT /api/core/application/vendors) and how to delete the application (DELETE /api/core/application/vendors). The hypermedia (the links) serve as the engine of application state (moving the caller through the different states of the application).

Chapter 2: Using the RESTful API

Components of a RESTful API Call

The RSA Archer RESTful API can be used in a number of ways. An example is to use a text editing tool like Notepad to create a header and a wrapper.

The following table details the various components of a RESTful API call within RSA Archer.

Component	Description	Sample Code
Call the API	Sets an HTTP Request.	`Sub CallTheArcherAPIFromVBS()` `End Sub`
Library	Uses HTTP by using a library.	`Set MyRequest = CreateObject ("winHttp.WinHttpRequest.5.1")`

Component	Description	Sample Code
Proxy	Tells the call where to send the HTTP Request to.	`MyRequest.setProxy 2, "127.0.0.1:8888"`
Request	Opens a request to send an API call to the server.	`MyRequest.Open "POST", "http://qeauto-web05.archereng.lab.emc.com/archer/api/core/security/login"`
Header	Tells the server what to do with the API call once it is received.	`MyRequest.setRequestHeader "Accept", "application/json,text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"` `MyRequest.setRequestHeader "Content-Type", "application/json"`

Component	Description	Sample Code
Body	The contents of the API call.	`requestBody = "{""InstanceName"":""Archer1,""Username"":""sysadmin"",""UserDomain"":"""",""Password"":""Archer2005""}"`
Submission	A call to submit the body to the server.	`MyRequest.send requestBody`

Response	Presenting the output from the server.	`Response = MyRequest.ResponseText`
		`token = Mid(Response, 48, 32)`
		`MsgBox token`

Example:

This particular sample finds the session token that the sysadmin user used to log in to the RSA Archer session.

```
CallTheRestAPIFromVBS
Sub CallTheArcherAPIFromVBS()
Set MyRequest = CreateObject(WinHttp.WinHttpRequest.5.1")
MyRequest.setProxy 2, "127.0.0.1:8888"
MyRequest.Open "POST", "http://qeauto-
web05.archereng.lab.emc.com/archer/api/core/security/login"
MyRequest.setRequestHeader "Accept",
"application/json,text/html,application/xhtml
+xml,application/xml;q=0.9,*/*;q=0.8"
MyRequest.setRequestHeader "Content-Type", "application/json"
```

```
requestBody =
"{""InstanceName"":""Archer1,""Username"":""sysadmin"",""UserDomai
n"":""",""Password"":""Archer2005""}"
MyRequest.send requestBody
Response = MyRequest.ResponseText
token = Mid(Response, 48, 32)
MsgBox token
End Sub
```

Determine Application GUIDs (REST)

Some methods require an application Globally Unique Identifier (GUID) as a parameter.

Determine the application GUID

1. Go to the Manage Applications page.
   a. From the menu bar, click  .
   b. Under Application Builder, click Applications.
2. In the Applications section, click the name of the application for which you want the GUID.
3. In the General Information section, the ID field contains the application GUID.

Determine System IDs (REST)

Some methods require a system ID as a parameter. Use the following procedure to find a Field ID, Application IDs, and Values List IDs.

Procedure

1. Go to the Manage Applications page.
   a. From the menu bar, click  .
   b. Under Application Builder, click Applications.
2. In the Applications section, click the name of the application that contains the field for which you want the ID.
3. Click the Fields tab.
4. In the Fields section, hover over the name of the appropriate field.
   The ID of the field displays in the bottom of the Manage Applications window.

Updating Locked Records (REST)

RSA Archer has an important feature that prevents the updating or altering of a locked record. A record becomes locked when a user has opened it in Edit mode for the purpose of modifying it.

However, it is important to note that records can be updated through the RESTful and Web APIs, as well as through data feeds, even when a user has locked them. The following are examples of typical APIs that can update user-locked records:

- PUT content (RESTful)
- UpdateRecord (Web Services)
- UpdateRecords (Web Services)

Chapter 3: Open Data Protocol (OData)

Open Data Protocol (OData)

The implementation of OData involves a custom processor that is accessible to each controller class derived from ArcherApiController. Because metadata and content are returned through different processes, the application of OData to these two different data sets varies. OData queries are normally passed on the request URI in a query string. This query is a supported use of OData in the RSA Archer API. All queries must be passed in the request body.

OData for MetaData

OData for MetaData - Projecting

Projecting is accomplished through the use of the OData $select keyword. This keyword allows the caller to specify which properties of the response object are to be included in the response.

The following example is of a valid OData select to return only the Name and Description properties of a response:

```
{"Value":"?$select=Name,Description"}
```

Projecting is supported by most of the URIs in the API.

OData for MetaData - Sorting

Sorting is accomplished through the use of the OData $orderby keyword. This keyword takes a property as an argument, together with ascending (asc) (the default) or descending (desc).

The following OData query will sort results in descending order by the value of the property boo:

```
{"Value":"?$orderby=boo desc"}
```

OData for MetaData - Paging and Limiting Results

Results can be paged using a combination of the OData keywords $top and $skip.

- $top sets the page size of the current request.
- $skip selects the correct page.

For example, assume that a particular request will return 1000 results. If you wanted the results to come back in 50 record pages and wanted to retrieve page 3, your OData query should look like this:

```
{"Value":"?$top=50&$skip=100"}
```

This query skips the first 100 records and returns the next 50, effectively returning page 3.

It is possible through configuration to set a page size on the server. Configuring the server for a usable page size limits the ability of the client to request a larger result set through an OData query. While this is limiting, it is a good idea for security reasons to configure a server side page size to the maximum size you want to allow. Doing so may prevent a scenario where a client inadvertently requests an unacceptably large number of records and monopolizes server resources with the request. It could also combat a malicious request intended to perpetrate a DOS by requesting a very large data set in a single page.

OData for Content

OData for Content - Projecting

Projecting the results of content requests is supported, but is subject to limitations similar to those of filtering. Projecting properties within a collection is not supported by OData. Projecting certain fields in a content record is not currently supported.

OData for Content - Paging and Limiting Results

$top, $skip, and $orderby are all fully supported when retrieving content for a reference field.

Chapter 4: Secure Use of HTTP Verbs

Secure Use of HTTP Verbs

A standard GET request is unencrypted. To avoid this inherent vulnerability, the API watches for and responds to the X-HTTP-Method-Override header. You can submit a request with the HTTP verb POST and the override header set to GET.

An example of the correct format of this header is as follows:

```
X-Http-Method-Override: GET
```

This request can contain additional data in the body, rather than in a query string. A handler early in the processing pipeline watches for this header and unpacks it. If the override verb is supported, the verb is changed and the request is passed along the processing pipeline.

A standard use of the GET request provides the additional information needed to fulfill the request in the query string. This data is plainly readable in web logs, as well as by anyone monitoring the HTTP communication.

Additional data could be placed in the request body, but is a non-standard use of a GET request and is not supported.

HTTP verb mapping

Mapping	HTTP Verb	HTTP Request	HTTP Response Type
Retrieve/Get	GET	Record	Record Set / Collection
Create	POST	Collection	Record Set / Collection
Update	PUT	Collection	Record Set / Collection
Delete	DELETE	Record	Record Set / Collection

Request verbs and actions

Request Verb	Action	Example
GET	Select vendor 123	GET /api/core/application/123
GET	Select all vendors	GET /api/core/application/

Request Verb	Action	Example
POST	Insert a new vendor	POST /api/core/application/
PUT	Update vendor 123	PUT /api/core/application/123
DELETE	Delete vendor 123	DELETE /api/core/application/123

Chapter 5: Mobile API

Mobile API Basics

The Mobile API is an HTTP service with a simplified interface similar to the Remote Procedure Call protocol.

Category	Description
Mobile Managers	Mobile API must contain business logic and API controllers do not contain business logic. Mobile Managers are located in a sub-folder in the RsaArcher/Api project. The managers are interface-based and make calls to managers from Common.
Assumptions	Calls to the Mobile API are secured using the same mechanisms as the RESTful API. Each call must include a session cookie or a properly formatted authorization header.
Routing and URIs	The Mobile API is contained in the RsaArcher/Api project in the RsaArcher/Api solution. The controllers are located in the Mobile/Controllers sub-folder. A full list of supported operations is found in the specification section of the API documentation.
Security	Mobile API includes: • Authentication. Authentication for the Mobile API is handled through the security controllers of the RESTful API. • Secure Use of HTTP Verbs. Like RESTful API, a standard use of the GET request provides the additional information needed to fulfill the request in the query string. GET requests that require additional input are sent as POST requests with the X-HTTP-Method-Override header set to GET.
Mobile API Calls	Synchronization (Sync Controller)

Synchronization (Sync Controller)

The Synchronization segment is accessed through the Sync Controller. Actions taken on a resource determine the type of action that can be performed.

Segment	Operation	URI	Arguments	Returns
StartSyncDownloadProcess	Begin a sync download process.	/sync/startsyncdownloadprocess	StartSyncRequest syncRequest	WebApiRequestResult<int>
CheckSyncDownloadProcessStatus	Check the status of an existing sync download process.	/sync/checksyncdownloadprocessstatus		WebApiRequestResult<JobStatusType>
FinishSyncDownloadProcess	Complete an existing sync download process and download the resulting zip file.	/sync/finishsyncdownloadprocess		HttpResponseMessage

Segment	Operation	URI	Arguments	Returns
ReportSyncDownloadProcess	Report that the download succeeded and update the sync status record to reflect that the zip file has been downloaded.	/sync/reportsyncdownloadsuccess		WebApiRequestResult<bool>
UpdateContent	Update an existing content record.	/sync/updatecontent	ContentSyncWrapper contentWrapper	WebApiRequestResult<int>

StartSyncDownloadProcess

The StartSyncDownloadProcess segment starts a mobile sync that queues an asynchronous job to generate metadata and content databases. This call returns a SyncProcessID that can be used to check the status of the sync process through a call to CheckSyncDownloadProcessStatus. When the sync download job is complete, obtain the resulting zip file through a call to FinishSyncDownloadProcess using the same SyncProcessID. You can use the SyncProcessID in other sync methods.

Request

```
POST http://RsaArcher/api/mobile/sync/startsyncdownloadprocess
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
"UniqueDeviceId":"abc123",
"KnownContentIds":[1, 2, 3]
}
```

Response Example

```
{
"RequestedObject":"995:
}
```

CheckSyncDownloadProcessStatus

The CheckSyncDownloadProcessStatus segment checks the status of a mobile sync job by SyncProcessId. When the job is complete, obtain the resulting zip file using that same SyncProcessId through a call to FinishSyncDownloadProcess.

This method must be called by the user on the device that originated the sync process. If the method is not called from the correct device, a 401 error occurs.

Request

```
POST
http://RsaArcher/api/mobile/sync/checksyncdownloadprocessstatus
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
"UniqueDeviceId":"abc123",
"SyncProcessId":995
}
```

Response Example

```
{
"RequestedObject":"3"
}
```

The following table describes the response value from 1 to 4 based on the state of the synchronization process:

Value	Synchronization Process
1	Download started
2	Download proceeding
3	Download finished
4	Download failed

FinishSyncDownloadProcess

The FinishSyncDownloadProcess segment obtains the zip file by SyncProcessId after the sync download job is completed.

This method must be called by the user on the device that originated the sync process or a 401 error occurs.

Request

```
POST http://RsaArcher/api/mobile/sync/finishsyncdownloadprocess
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
"UniqueDeviceId":"abc123",
```

```
"SyncProcessId":995
}
```

The response from this call is a byte array containing the database files output in a zipped file.

ReportSyncDownloadSuccess

When the download of the content and metadata zip file is completed, the ReportSyncDownloadSuccess method updates the job sync status table with the download date. If this call is not made, the system does not consider the contents of that zip file to have been successfully downloaded, but does allow the contents to be included in subsequent sync files.

Request

```
POST http://RsaArcher/api/mobile/sync/reportsyncdownloadsuccess
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
"UniqueDeviceId":"abc123",
"SyncProcessID":[995]
}
```

UpdateContent

This segment updates an existing content record when sending assessments back to RSA Archer through mobile synchronization.

When the subform content record is updated, subformFieldId must be passed in the body of the request in order to properly associate the content with the correct subform field.

Request

```
PUT http://RsaArcher/api/mobile/sync/updatecontent
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id=session token ID from login
```

```
Content-Type: application/json
```

Request Body

```
{
"LevelId" : 9,
"FieldContents" :
  {
  "3204" :
    {
    "Type" : 2,
    "Value" : 0.0,
    "FieldId" : 3204
    },
  "3202" :
    {
    "Type" : 4,
    "Value" :
      {
      "ValuesListIds" : [2405],
      "OtherText" : null
      },
    "FieldId" : 3202
    },
  "10066" :
    {
    "Type" : 23,
    "Value" : [150622],
    "FieldId" : 10066
    }
  }
}
```

Chapter 6: RESTful API Segments

The RSA Archer RESTful API is divided into the following functional segments:

Authentication

Content

Data Feed

Metadata

Inputs of a RESTful API Segment

RSA Archer RESTful API segments use many different fields within inputs. This section explains how to format those inputs in an API call.

Validation

Double check your input carefully before creating or updating records. Invalid inputs are ignored despite an apparently successful API response of True for IsSuccessful. Validation messages are not provided.

Examples of invalid inputs are incorrect IP addresses and non-existent ContentIds.

Optional Tags

Inputs can become lengthy and difficult to scan quickly. To avoid confusion, RSA recommends that you use placeholders within your inputs to leave breadcrumbs through the logic. The examples in this document include an optional input value, Tag. This placeholder allows you to easily follow the logic of each input. The system ignores placeholder inputs during command execution.

The following is an example of a tag used in a date input.

```
"Tag":"Date of Birth"
```

Note: Do not use this optional field for the IP input.

Null Value

Null indicates a condition where values are missing. There are times where you want to reset an existing value to null. In most cases, simply omit the input field from the API command to pass a null value into the record. Some input fields require specific values to indicate a null condition. Those input fields are:

- Date
- Values List
- Users/Groups List

Input Fields

The following table details the inputs of a RESTful API segment within RSA Archer.

Input Field	Segment Used In	Sample Code

Text	Create/Update Content,
	Post Attachment,
	Execute Data Feed,
	Add Group Member,
	Create/Update Group,
	Create/Update Role,
	Create/Update a Security Parameter,
	Change User Password,
	Create/Update User
		"9402"
		{
		"Type": 1,
		"Tag":"TextString",
		"Value": "ABCDEFG hijk LMNOP qrstuvwxyz ÃÄÇËÑßÜ áòëæñü 信息報이ÜΦ",
		"FieldId":9402
		}
		
		"9403"
		{
		"Type": 1,
		"Tag":"TextHTML",
		"Value": " 1234 Main Street
		Anytown, AA 12345 ",
		"FieldId":9403
		}
		
		Note: Use Text as an input for any string or HTML text block.

Input Field	Segment Used In	Sample Code

Numeric	Get Content by Content ID and Field ID,
	Create/Update Content,
	Add/Remove Group Member,
	Add/Remove Group to Role,
	Update Group,
	Create/Update a Security Parameter,
	Add User to Role,
	Add User to User Group,
	Change User Password,
	Deactivate/Delete/Update User,
	Get Contact Information for a User,
	Remove User from User Group,
	Remove User from Role
		`"1234"`
		`{`
		`"Type": 2,`
		`"Tag":"Numeric",`
		`"Value": 123450.6789,`
		`"FieldId":1234`
		`}`
		Note: Use Numeric as an input for numbers.

Input Field	Segment Used In	Sample Code

Date	Create/Update Content,
	Create/Update a Security Parameter,
	Create/Update User
		"123451"
		{
		"Type": 3,
		"Tag":"Date with Time",
		"Value": "10/21/1956 11:59AM",
		"FieldId":123451
		}
		
		"123452"
		{
		"Type": 3,
		"Tag":"Date",
		"Value": "12/27/2016",
		"FieldId":123452
		}
		
		Note: Use Date as an input for any Date or Date with a time.
		
		"123453"
		{
		"Type": 3,
		"Tag":"Null (Date)",
		"Value": "0",
		"FieldId":123453
		}
		
		Note: To set as Null, use "0" as the input.

Input Field	Segment Used In	Sample Code

Values List	Create/Update Content,
	Create/Update User
		"34564"
		{
		"Type": 4,
		"Tag":"Severity (Values List)",
		"Value":
		{
		"ValuesListIds":[2,3,4],
		"OtherText":"Severity"
		}
		"FieldID":34564
		}
		
		"34563"
		{
		"Type": 4,
		"Tag":"Null (Values List)",
		"Value": {0}
		"FieldID":34563
		}

External Links	Create/Update Content,
	Create/Update User
		"11234"
		{
		"Type": 7,
		"Tag":"External Links",
		"Value":
		[
		{
		"Name": "RSA",
		"URL":"https://www.rsa.com"
		},
		{
		"Name":"Archer Community",
		"URL":"https://community.rsa.com/comm
		unity/products/archer-grc"
		}
		],
		"FieldId": 11234
		}

Input Field	Segment Used In	Sample Code

Users/Groups List		"12234"
		{
		"Type": 8,
		"Tag":"Users or Group List",
		"Value":
		{
		"UserList":
		[
		{
		"ID" : 19
		},
		{
		"ID" : 20
		}
		],
		"GroupList":
		[
		{
		"ID" : 1
		}
		]
		},
		"FieldId": 12234
		}

Input Field	Segment Used In	Sample Code

Record Permissions		"13877"
		{
		"Type": 8,
		"Tag":"Record Permissions",
		"Value":
		{
		"UserList":
		[
		{
		"ID" : 190
		},
		{
		"ID" : 191
		}
		],
		"GroupList":
		[
		{
		"ID" : 19
		}
		]
		},
		"FieldId": 13877
		}

Input Field	Segment Used In	Sample Code

Cross-Reference	Create/Update Content,
	Create/Update User
		"12234"
		{
		"Type": 9,
		"Tag":"Cross-Reference",
		"Value":
		[
		{
		"ContentID":205522
		},
		{
		"ContentID":205643
		},
		{
		"ContentID":205783
		},
		],
		"FieldId" : 12234
		}

Attachment	Create/Update Content,
	Create/Update User
		"12334"
		{
		"Type": 11,
		"Tag":"Attachments",
		"Value":
		[
		1234
		],
		"FieldId" : 12334
		}
		
		Note: The value in this input is the Attachment ID, which is returned by the *Post Attachment by Bytes* and *Post attachment from multipart form data* segment. See Content for more details on these segments.

Input Field	Segment Used In	Sample Code

Image	Create/Update Content,
	Create/Update User
		<pre>"12334"
		{
		  "Type": 12,
		  "Tag":"Screenshot (Image)",
		  "Value":
		  [
		  2234
		  ],
		  "FieldId": 12334
		}</pre>
		Note: An Image is a type of attachment, so they are treated the same way. The difference is the Type number. Images are Type 12.

Matrix	Create/Update Content,
	Create/Update User
		<pre>"15674"
		{
		  "Type": 16,
		  "Tag":"Matrix",
		  "Value":
		  [
		  {
		  "RowId":63800,
		  "ColumnId":63823
		  }
		  ],
		  "FieldId": 15674
		}</pre>

IP Address	Create/Update Content,
	Create/Update User
		<pre>"12334"
		{
		  "Type": 19,
		  "IpAddressBytes":"127.0.0.1",
		  "FieldId": 12334
		}</pre>
		Note: Do not include any additional fields. If there are additional fields, this input fails without returning a message.

Input Field	Segment Used In	Sample Code

Related Records	Create/Update Content,
	Create/Update User
		"18975"
		{
		"Type": 23,
		"Tag":"Related Records",
		"Value":
		[
		345155,
		234512
		],
		"FieldId": 18975
		}
		Note: The Value contains a set of content IDs that were assigned when the Related Record was originally created.

Sub-Form	Create/Update Content,
	Create/Update User
		"18975"
		{
		"Type": 24,
		"Tag":"Subforms",
		"Value":
		[
		345155
		],
		"FieldId": 18975
		}
		Note: The Value contains a set of content IDs that were assigned when the Sub-Form was originally created.

Authentication

SessionContext can be obtained by passing valid credentials to the Login method of the Security controller. This SessionContext contains a string called SessionToken which appears in every subsequent request in a custom Authentication header.

Example:

```
Authorization: Archer session-
id="D45BB616F3F564B12199F8198522042D"
```

Note: Insert the session token ID from your login. In this documentation, the Archer session-id is represented as "session token ID from login."

Because SessionContext is obtained through the AuthenticationManager, each call must conform to the normal restrictions of their associated security parameters in RSA Archer. Every request of the API must contain a valid Authentication header using the RSA Archer authentication scheme.

The Authentication segment is accessed through the Security controller and includes the following services:

Login

Login creates an RSA Archer session using the specified credentials on the specified instance. The API request returns a serialized representation of a SessionContext object.

Request

```
POST http://RsaArcher/api/core/security/login
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Content-Type: application/json
```

Request Body

```
{
"InstanceName":"v5.0",
"Username":"sysadmin",
"UserDomain":"",
"Password":"Archer123"
}
```

Response Example

```
{
“SessionToken”:” C204E18D0ED58E288533F39C455A36E8”
}
```

Logout

Logout calls a service that terminates the specified session. If the session in question is already terminated or is invalid, the method still returns true. The user associated with the session in the authorization header must have the necessary permissions to terminate the session of another user.

Normally, the session token ID is the same for both Login and Logout, assuming it is the same user. If a user has permission to log other users out, the Logout session token ID is for the user logging out.

Request

```
POST http://RsaArcher/api/core/security/logout
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
"Value":"session token ID to logout"
}
```

Content

The Content segment is accessed through the Content controller. Actions taken on a resource determine the type of action that can be performed.

The following table describes which actions can be taken on the content all resource.

Resource	URI	$filter	$select	$top	$skip	$orderby
content - by id	/content/contentid	❌	✅	❌	❌	❌
content - by reference field ID	/content/referencefield/referencefieldid	✅	✅	✅	✅	✅
content - by content ID and field ID	/content/fieldcontent	✅	✅	✅	✅	✅

Resource	URI	$filter	$select	$top	$skip	$orderby
content - expose history log data	/content/history/ID	✅	✅	✅	✅	✅
content - post	/content	❌	❌	❌	❌	❌
content - put	/content	❌	❌	❌	❌	❌
attachment - get	/content/attachment/attachmentid	❌	❌	❌	❌	❌
attachment - post using bytes	/content/attachment	❌	❌	❌	❌	❌
attachment - post using multipart form data	/content/multipartattachment	❌	❌	❌	❌	❌

Get content by ID

The Get content by ID resource retrieves a content record by the specified ID.

Request

```
POST http://RsaArcher/api/core/content/contentid
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get content by reference field ID

The Get content by reference field ID resource retrieves all the content records by the specified

reference field ID. If the reference field is a target cross reference for a questionnaire, this URI returns all content in the target application. If the reference field is a regular cross-reference or a related-record field, this URI returns the record lookup content.

Request

```
POST
http://RsaArcher/api/core/content/referencefield/referencefieldid
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get content by content ID and field ID

The Get content by content ID and field ID resource retrieves field content by content ID and field ID in the current RSA Archer instance.

Request

```
POST http://RsaArcher/api/core/content/fieldcontent
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,applicatio
n/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
  "FieldIds":[111,112,113],
  "ContentIds":[700,701,702]
}
```

Expose History Log Data

The Expose History Log Data resource retrieves content on every history log field in the application. If there is no content in the History Log field, the response for that field is empty. If no history log fields in the application contain relevant content, the response for the application is empty.

Request

```
POST http://localthost/Archer/api/core/content/history/ID
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Post content

The Post content resource saves a content record. The content must be provided in the body of the request in the format specified by the content-type header. Attempting to update an existing content record using POST results in a 403 - Forbidden error.

Request

```
POST http://RsaArcher/api/core/content
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
  "Content":
  {
    "LevelId" : 9,
    "FieldContents" :
    {
      "3204" :
      {
        "Type" : 2,
        "Value" : 0.0,
        "FieldId" : 3204
      },
      "3202" :
      {
        "Type" : 4,
        "Value" :
```

```
      {
        "ValuesListIds" : [2405],
        "OtherText" : null
      },
      "FieldId" : 3202
    },
    "10066" :
    {
      "Type" : 23,
      "Value" : [150622],
      "FieldId" : 10066
    }
  }
},
"SubformFieldId" : 123
}
```

The request body shows an example of a Content record serialized in JSON format. This content record contains three fields:

- Numeric field
- Values list field
- Cross-reference field

The Content Wrapper can also contain a SubformFieldId that must be passed when subform content is being saved. Otherwise, this parameter can be omitted.

- If the save is successful, the requested object returned from this URI includes the ID of the content.
- If the save is not successful, a validation messages explaining the nature of the failure is displayed.

Put content

The Put content resource updates an existing content record. The content must be provided in the body of the request in the format specified by the content-type header. Attempting to insert a new content record using PUT results in a 403 - Forbidden error.

Request

```
PUT http://RsaArcher/api/core/content
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{
  "Content":
  {
    "Id": 123,
    "LevelId" : 9,
    "FieldContents" :
    {
      "3204" :
      {
        "Type" : 2,
        "Value" : 0.0,
        "FieldId" : 3204
      },
      "3202" :
      {
        "Type" : 4,
        "Value" :
        {
          "ValuesListIds" : [2405],
          "OtherText" : null
        },
        "FieldId" : 3202
      },
      "10066" :
      {
        "Type" : 23,
        "Value" : [150622],
        "FieldId" : 10066
      }
    },
    "Version": "44"
  },
  "SubformFieldId":123
}
```

The request body shows an example of a Content Wrapper, which contains a content record serialized in JSON format. This content record contains three fields:

- A numeric field.
- A values list field.
- A cross-reference field.

The Content Wrapper can also contain a SubformFieldId that must be passed when subform content is being saved. Otherwise, this parameter can be omitted.

- If the save was successful, the requested object returned from this URI will include the id of the content.
- If the save was not successful, a validation messages explaining the failure is displayed.

Get attachment

The Get attachment resource retrieves an attachment from the RSA Archer file repository. The actual object returned is an AttachmentWrapper, which has two properties:

- AttachmentName
- AttachmentBytes

The bytes are represented as a Base64 encoded string.

Request

```
POST http://RsaArcher/api/core/content/attachment/*attachmentid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Post attachment from bytes

The Post attachment from bytes resource accepts a serialized AttachmentWrapper with the AttachmentName and AttachmentBytes properties. If either of these properties is not present, the post fails with a validation message indicating that the request was badly formatted. AttachmentBytes must contain a Base64 encoded string representation of the bytes of the attachment file. On success, the API returns the newly assigned id of the attachment file.

Request

```
POST http://RsaArcher/api/core/content/attachment
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
"Attachment Bodie"
{
  "AttachmentName":"Bodie.jpg",
  "AttachmentBytes":"/9j/4AAQSkZJRgABAgAAAQABAAD/4QDmRXhpZgAASUkqA
  AgAAAAFABIBAwABAAAAAQAAADEBAgAcAAAASgAAADIBAgAUA
  AAAZgAAABMCAwABAAAAAQAAAGmHBAABAAAAegAAAAAAAABBQ0QgU3lzdGVtcyBEa
  WdpdGFsIE"
}
```

Post attachment from multipart form data

The Post attachment from multipart form data resource accepts multipart form data containing an attachment and attachment name. These values must be included in content headers with the AttachmentName and AttachmentBytes names. If either of these content headers is not present, the post fails with a validation message indicating that the request was badly formatted. AttachmentBytes must contain a Base64 encoded string representation of the bytes of the attachment file. If the request is being posted from a form, this request is handled by the browser when the request is submitted. On success, the API returns the newly assigned ID of the attachment file.

Request

```
POST http://RsaArcher/api/core/content/multipartattachment
```

Request Header

```
Content-Type: multipart/form-data; boundary=----------------------
---acebdf13572468
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
```

Request Body

```
--------------------------acebdf13572468
```

```
Content-Disposition: form-data; name="AttachmentName"
MyPhoto.jpg
--------------------------acebdf13572468
Content-Disposition: form-data; name="AttachmentBytes";
filename="Account.jpg"
Content-Type: image/jpeg
LS0tLS0tV2ViS2l0Rm9ybUJvdW5kYXJ5QmZMRkloZUhkdzluWDhNeA0KQ29udGVudC
1EaXNwb3NpdGlvbjogZm9ybS1kYXRhOyBuYW1lPSJfX1ZJRVdTVEFURSINCg0KL3dF
UER3VUxMVEV3TVRrd
09EazBNallQWkJZQ1pnOWtGZ0lDQXc4V0J
--------------------------acebdf13572468--
```

Data Feed

The Data Feed segment is accessed through the Data Feed controller. Actions taken on a resource determine the type of action that can be performed.

The following table describes which actions can be taken on the content all resource.

Resource	URI	$filter	$select	$top	$skip	$orderby
Execute Data Feed	/datafeed/executedatafeed	❌	❌	❌	❌	❌
Get History Message for History	/datafeed/historymessage/historyId	✅	✅	✅	✅	✅
Get Recent Run Detail	/datafeed/getrecentrundetail	❌	✅	❌	❌	❌
Get Run History	/datafeed/history	✅	✅	✅	✅	✅

Execute Data Feed

The Execute Data Feed resource executes a data feed and any data feeds that may be referenced. The IsReferenceFeedsIncludedflag must be true before any referenced data feeds can run. If you want to run a single data feed without any referenced feeds, the flag must be set to false. The session token ID must be for a user who has access rights to run data feeds.

Request

```
POST http://RsaArcher/api/core/datafeed/execution
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,applicatio
n/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Request Body

```
{"DataFeedGuid":"4685367E-E76E-4580-BDEC-
83E98FF48A50","IsReferenceFeedsIncluded":false}
```

Get History Message for a History

The Get History Message for a History resource retrieves a history message for a specific history ID.

Request

```
POST http://RsaArcher/api/core/datafeed/historymessage/historyId
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,applicatio
n/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get Recent Run Detail for a Data Feed

The Get Recent Run Detail for a Data Feed resource retrieves history information for the most recent data feed, including execution time, number of records created, updated or deleted, status, and whether the feed succeeded or failed.

Request

```
POST http://RsaArcher/api/core/datafeed/history/recent
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,applicatio
n/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Request Body

```
{"Guid":"4685367E-E76E-4580-BDEC-83E98FF48A50"}
```

Get Run History for a Data Feed

The Get Run History for a Data Feed resource retrieves history information for a data feed, including execution time, number of records created, updated or deleted, status, and whether the feed succeeded or failed.

Request

```
POST http://RsaArcher/api/core/datafeed/history
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,applicatio
n/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Request Body

```
{"Guid":"4685367E-E76E-4580-BDEC-83E98FF48A50"}
```

Metadata

The Medadata segment is accessed through the System controller. Actions taken on a resource determine the type of action that can be performed.

The following table provides the actions that can be taken on each resource, including the URI and object type of the resource.

Segment	Resource	URI	$filter	$select	$top	$skip	$orderby
Application	application - by id	/system/application/applicationid	❌	✅	❌	❌	❌
	application - all	/system/application	✅	✅	✅	✅	✅
	application version	/system/applicationinfo/version	❌	❌	❌	❌	❌
Event	event action - by id	/system/eventaction/eventactionid	❌	✅	❌	❌	❌

Segment	Resource	URI	$filter	$select	$top	$skip	$orderby
	event action - by event rule	/system/eventaction/eventrule/eventruleid	✅	✅	✅	✅	✅
	event rule - by ID	/system/eventrule/eventruleid	❌	✅	❌	❌	❌
	event rule - by level	/system/eventrule/level/levelid	✅	✅	✅	✅	✅
Field	field content - by field ID	/system/content/fieldcontent	✅	✅	✅	✅	✅
	field definition - by ID	/system/fielddefinition/fieldid	❌	✅	❌	❌	❌
	field definition - by application ID	/system/fielddefinition/application/*applicationid*	✅	✅	✅	✅	✅
	field definition - by level	/system/fielddefinition/level/levelid	✅	✅	✅	✅	✅
	field definition - by level for ValuesList	/system/fielddefinition/level/levelid/valueslist	✅	✅	✅	✅	✅
	field display - by level	/system/fielddisplay/level/levelid	✅	✅	✅	✅	✅

Segment	Resource	URI	$filter	$select	$top	$skip	$orderby
	field id - by eventrule filter by level	api/system/field/eventrulefilter/level/levelId	❌	❌	❌	❌	❌
Group	group - by ID	/system/group/groupid	❌	✅	❌	❌	❌
	group - by user	/system/group/user/userid	✅	✅	✅	✅	✅
	group - all	/system/group	✅	✅	✅	✅	✅
	group hierarchy - all	/system/grouphierarchy	✅	✅	✅	✅	✅
	group - create	/system/group	❌	❌	❌	❌	❌
	group - update	/system/group	❌	❌	❌	❌	❌
	group - delete	/system/group/*groupid*	❌	❌	❌	❌	❌
	group membership - all	/system/groupmembership	✅	✅	✅	✅	✅
	group - add member	/system/groupmember	❌	❌	❌	❌	❌
	group - remove member	/system/groupmember	❌	❌	❌	❌	❌
	group - add to role	/system/rolegroup	❌	❌	❌	❌	❌

Segment	Resource	URI	$filter	$select	$top	$skip	$orderby
	group - remove from role	/system/rolegroup	❌	❌	❌	❌	❌
Level	level - by id	/system/level/levelid	❌	✅	❌	❌	❌
	level - by module	/system/level/module/moduleid	✅	✅	✅	✅	✅
	level - by reference field	/system/level/referencefield/fieldid	✅	✅	✅	✅	✅
	level - all	/system/level	✅	✅	✅	✅	✅
	level layout - by level id	/system/levellayout/levelid	❌	✅	❌	❌	❌
Questionnaire	questionnaire - by id	/system/questionnaire/questionnaireid	❌	✅	❌	❌	❌
	questionnaire - all	/system/questionnaire	✅	✅	✅	✅	✅
	questionnaire rule - by id	/system/questionnairerule/id	❌	✅	❌	❌	❌
	questionnaire rule - by level	/system/questionnairerule/level/levelid	✅	✅	✅	✅	✅
Role	role - all	/system/role	✅	✅	✅	✅	✅
	role - by user ID	/system/role/user/*userid*	✅	✅	✅	✅	✅
	role - memberships	/system/role	✅	✅	✅	✅	✅

Segment	Resource	URI	$filter	$select	$top	$skip	$orderby
	role - create	/system/role	❌	❌	❌	❌	❌
	role - update	/system/role	❌	❌	❌	❌	❌
	role - delete	/system/role/*roleid*	❌	❌	❌	❌	❌
Security	security parameter - by id	/system/securityparameter/securityparameterid	❌	✅	❌	❌	❌
	security parameter - all	/system/securityparameter	✅	✅	✅	✅	✅
	security parameter - create	/system/securityparameter	❌	❌	❌	❌	❌
	security parameter - update	/system/securityparameter	❌	❌	❌	❌	❌
	security parameter - delete	/system/securityparameter/*securityparameterid*	❌	❌	❌	❌	❌
Selected User Group	selected user group - by field id	/system/usergroupselection/fieldid	✅	✅	✅	✅	✅
Subform	subform - by id	/system/subform/subformid	❌	✅	❌	❌	❌
User	user - by id	/system/user/userid	❌	✅	❌	❌	❌
	user - by group	/system/user/group/groupid	✅	✅	✅	✅	✅

Segment	Resource	URI	$filter	$select	$top	$skip	$orderby
	user - all	/system/user	✅	✅	✅	✅	✅
	user - create	/system/user	❌	❌	❌	❌	❌
	user - update	/system/user	❌	❌	❌	❌	❌
	user - delete	/system/user/*userid*	❌	❌	❌	❌	❌
	user - activate	/system/user/status/active/*userid*	❌	❌	❌	❌	❌
	user - deactivate	/status/inactive/*userid*	❌	❌	❌	❌	❌
	user - change password	/system/userpassword	❌	❌	❌	❌	❌
	user - add to group	/system/usergroup	❌	❌	❌	❌	❌
	user - remove from group	/system/usergroup	❌	❌	❌	❌	❌
	user - add to role	/system/userrole	❌	❌	❌	❌	❌
	user - remove from role	/system/userrole	❌	❌	❌	❌	❌
	user contacts-all	/system/usercontact	✅	✅	✅	✅	✅

Segment	Resource	URI	$filter	$select	$top	$skip	$orderby
	user contacts - by user ID	/system/usercontact/*userid*	✅	✅	✅	✅	✅
Values List	values list definition - by id	/system/valueslist/id	❌	✅	❌	❌	❌
	values list value - by values list id (flat)	/system/valueslistvalue/flat/valueslist/valueslistid	❌	❌	❌	❌	❌
	values list value - by values list id (hierarchical)	/system/valueslistvalue/valueslist/valueslistid	❌	❌	❌	❌	❌

Metadata Application

The Application segment is accessed through the System controller, which includes the following resources:

Get all applications

The Get all applications resource retrieves metadata for all applications.

Request

```
POST http://RsaArcher/api/core/system/application
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
```

```
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get application by ID

The Get application by ID resource retrieves metadata for the application by the application ID.

Request

```
POST http://RsaArcher/api/core/system/application/*applicationid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get application version

The Get application version resource retrieves the version number for the RSA Archer instance against which the API is running.

Request

```
POST http://Archer/api/core/system/applicationinfo/version
(http://Archer/api/core/system/applicationinfo/version)
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Metadata Event Actions and Rules

The Event Actions segment is accessed through the System controller, which includes the following resources:

Get event action by ID

The Get event action by ID resource retrieves the data driven event action of the specified ID.

Request

```
POST http://RsaArcher/api/core/system/eventaction/*eventactionid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get event action by event rule ID

The Get event action by event rule ID resource retrieves the data driven event actions associated with the event rule by the specified event rule ID.

Request

```
POST
http://RsaArcher/api/core/system/eventaction/eventrule/*eventrulei
d*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

The Event Rules segment is accessed through the System controller, which includes the following resources:

Get event rule by ID

The Get event rule by ID resource retrieves the data driven event rule by the specified ID.

Request

```
POST http://RsaArcher/api/core/system/eventrule/*eventruleid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
```

```
Content-Type: application/json
X-Http-Method-Override: GET
```

Get event rule by level ID

The Get event rule by level ID resource retrieves the data driven event rule associated with the level by the level ID.

Request

```
POST http://RsaArcher/api/core/system/eventrule/level/*levelid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Metadata Field

Some text fields allow for both plain text and rich text. For example, an address can be formatted as one line in plain text or multiple lines in rich text.

The following table shows which fields accept both plain text and rich text.

Metadata	Field
Group	Description
Role	Description
Security Parameter	Description
User	Address
User	AdditionalNote

Supported field types for RESTful API

The following table shows supported field types.

Field Type ID	Field Type	Syntax
1	Text	FieldType.Text
2	Numeric	FieldType.Numeric
3	Date	FieldType.Date
4	Values List	FieldType.ValuesList
6	TrackingID	FieldType.TrackingID
7	External Links	FieldType.ExternalLinks
8	Users/Groups List	FieldType.UsersGroupsList
9	Cross-Reference	FieldType.CrossReference
11	Attachment	FieldType.Attachment
12	Image	FieldType.Image
14	Cross-Application Status Tracking (CAST)	FieldType.CastScoreCard
16	Matrix	FieldType.Matrix
19	IP Address	FieldType.IpAddress
20	Record Status	FieldType.RecordStatus
21	First Published	FieldType.FirstPublishedDate
22	Last Updated Field	FieldType.LastUpdatedField
23	Related Records	FieldType.RelatedRecords
24	Sub-Form	FieldType.SubForm
25	History Log	FieldType.HistoryLog
26	Discussion	FieldType.Discussion
27	Multiple Reference Display Control	FieldType.MultipleReferenceDisplayControl
28	Questionnaire Reference	FieldType.QuestionnaireReference

Field Type ID	Field Type	Syntax
29	Access History	FieldType.ContentAccessHistory
30	Voting	FieldType.Voting
31	Scheduler	FieldType.Scheduler
1001	Cross-Application Status Tracking Field Value	FieldType.CrossApplicationStatusTrackingFieldValue

The Field Definition and Field Display segments are accessed through the System controller, which includes the following resources.

Get field definition by application ID

Retrieves field definition for an application ID in the current RSA Archer instance.

Request

```
GET
http://localhost/Archer/api/core/system/fielddefinition/application/*applicationid*
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,application/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Get field definition by field ID

Retrieves the field definition of the specified field ID. Although this field definition will be of a specific derived type, (for example, ValuesListFieldDefinition, TextFieldDefinition, and others) the field definition will be returned as the base class FieldDefinition. Only properties of the base class can be used in OData query strings. To filter or select by properties of a specific type of field definition, use the URI dedicated to that field definition type. For example, to filter by properties of ValuesListFieldDefinition, use the URI /system/fielddefinition/level/levelid/valueslist.

Request

```
POST
http://RsaArcher/api/core/system/fielddefinition/*fielddefinitionid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get field definition by level ID

Retrieves all field definitions for the level by the specified level ID.

Request

```
POST
http://RsaArcher/api/core/system/fielddefinition/level/*levelid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get values list field definition by level ID

Retrieves the field definition by the specified level ID. Because field definitions, regardless of their derived type, are all returned as the base class FieldDefinition, only properties of the base class can be used in OData query strings. This method provides filtering by properties specific to ValuesListFieldDefinition.

Request

```
POST
http://RsaArcher/api/core/system/fielddefinition/level/*levelid*/v
alueslist
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get field display by level ID

Retrieves field displays for the level by the specified level ID.

Request

```
POST http://RsaArcher/api/core/system/fielddisplay/level/*levelid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get field IDs by Event Rule filter by level ID

Retrieves a list of the field definitions that are referenced by an Event Rule filter on the specified level.

Request

```
GET
http://RsaArcher/api/core/system/field/eventrulefilter/level/*leve
lid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="session token ID from login"
Content-Type: application/json
```

Metadata Group

The Group segment is accessed through the System controller, which includes the following resources:

Add group member

The Add group member resource adds a group to another group. GroupId is the ID of a parent group and GroupMemberId is the ID of a child group. Both parent group and child group must exist.

Request

```
PUT http://RsaArcher/api/core/system/groupmember
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "GroupId":123,
  "GroupMemberId":789,
  "IsAdd":true
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":206
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Add group to role

Adds a group to an access role. Both group and access role must exist.

Request

```
PUT http://RsaArcher/api/core/system/rolegroup
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "GroupId":206,
  "RoleId":3,
  "IsAdd":true
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":206
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Create group

Creates a group. The name field is required in the request body.

Request

```
POST http://RsaArcher/api/core/system/group
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "Group":
        {
        "Name":"GroupA",
        "Description":"Group A description"
        },
  "ParentGroups":[1],
  "ChildGroups":[2],
  "ChildUsers":null
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":159
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Delete group

Deletes a group.

Request

```
DELETE http://RsaArcher/api/core/system/group/*groupid*
DELETE http://RsaArcher/api/core/system/group/206
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
None
```

Response Example

```
{
  "Links":[],
  "RequestedObject":206,
  "IsSuccessful":true,
  "ValidationMessages":null
}
```

Get all groups

Retrieves all groups.

Request

```
POST http://RsaArcher/api/core/system/group
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Request Body

```
None
```

Response Example


```
[
  {
    "Links":[],
    "RequestedObject":
      {
        "Id":206,
        "Name":"New Group Name",
        "DisplayName":"New Group Name",
        "Description":"<html><head></head><p style=\"margin:
        0px\">New Group Description</p></html>",
        "EveryoneFlag":false,
        "Guid":"6dc2e265-2796-4e3c-ab8e-549ba03422d8",
        "SystemFlag":false,
        "LdapFlag":false,
        "DomainId":null,
        "DistinguishedName":null,
        "DefaultHomeDashboardId":-1,
        "DefaultHomeWorkspaceId":-1,
        "UpdateInformation":
          {
          "CreateDate":"2016-09-21T18:04:02.643",
          "UpdateDate":"2016-09-21T18:06:51.027",
          "CreateLogin":1470,
          "UpdateLogin":1470
          }
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  }
]
```

Note: The above example is just one group. This command will bring up as many groups that exist in your environment.

Get group by ID

Retrieves a group by the specified group ID.

Request

```
GET http://RsaArcher/api/core/system/group/*groupId*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
none
```

Response Example


```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":208,
      "Name":"Updated Group Name",
      "DisplayName":"Updated Group Name",
      "Description":"<html><head></head><p style=\"margin:
      0px\">WAS: New Group Description</p></html>",
      "EveryoneFlag":false,
      "Guid":"4eec52f1-bc2f-4460-ac2b-e6e179b5744d",
      "SystemFlag":false,
      "LdapFlag":false,
      "DomainId":null,
      "DistinguishedName":null,
      "DefaultHomeDashboardId":null,
      "DefaultHomeWorkspaceId":null,
      "UpdateInformation":
        {
          "CreateDate":"2016-09-22T16:07:59.807",
          "UpdateDate":"2016-09-22T16:09:01.083",
```

```
          "CreateLogin":1470,
          "UpdateLogin":1230
        }
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Get group hierarchy

Retrieves the hierarchy for all groups. Using this hierarchy, the relationships between groups can be composed into a tree structure. A group hierarchy record consists of the group id, a related id, and a generation. The related id refers to a group that is an ancestor of the current group. The generation describes how far removed the current group is from that related group. A group is always in generation zero (0) for itself. A group is in generation 1 for its immediate parent.

Request

```
GET http://RsaArcher/api/core/system/grouphierarchy
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
None
```

Response Example

```
[
  {
    "Links":[],
    "RequestedObject":
      {
        "Id":2,
        "RelatedId":2,
        "Generation":0
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  },
```

```
  {
    "Links":[],
    "RequestedObject":
      {
        "Id":3,
        "RelatedId":3,
        "Generation":0
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  },
  {
    "Links":[],
    "RequestedObject":
      {
        "Id":4,
        "RelatedId":4,
        "Generation":0
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  }
]
```

Get all group memberships

Retrieves memberships for all groups. Membership includes all users and all parent groups for a group.

Request

```
GET http://RsaArcher/api/core/system/groupmembership
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
None
```

Request Response


```
[
  {
    "Links":[],
    "RequestedObject":
      {
        "GroupId":1,
        "UserIds":null,
        "ParentGroupIds":null
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  },
  {
    "Links":[],
    "RequestedObject":
      {
        "GroupId":2,
        "UserIds":null,
        "ParentGroupIds":[16]
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  },
  {
    "Links":[],
    "RequestedObject":
      {
        "GroupId":16,
        "UserIds":[1470],
        "ParentGroupIds":null
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  },
  {
    "Links":[],
    "RequestedObject":
      {
        "GroupId":17,
        "UserIds":null,
        "ParentGroupIds":[16]
      },
```

```
    "IsSuccessful":true,
    "ValidationMessages":[]
  },
  {
    "Links":[],
    "RequestedObject":
      {
        "GroupId":18,
        "UserIds":null,
        "ParentGroupIds":[16]
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  },
  {
    "Links":[],
    "RequestedObject":
      {
        "GroupId":19,
        "UserIds":[1355],
        "ParentGroupIds":[16]
      },
    "IsSuccessful":true,
    "ValidationMessages":[]
  }
]
```

Get groups by user

Retrieves all groups of which the specified user is a member.

Request

```
POST http://RsaArcher/api/core/system/group/user/*userId*
POST http://RsaArcher/api/core/system/group/user/1470
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Request Body

```
None
```

Response Example

```
[
  {
  "Links":[],
  "RequestedObject":
    {
    "Id":204,
    "Name":"Asset Catalog Administator",
    "DisplayName":"Asset Catalog Administator",
    "Description":"<html><head></head><p style=\"margin:
    0px\">The Asset Catalog Administrator group is for users who
    have full administrative access to the following core
    applications: Applications, Devices, Storage Devices,
    Technologies, Business Processes, Corporate Objectives,
    Products and Services, Information Assets, Facilities, and
    Contacts.</p></html>",
    "EveryoneFlag":false,
    "Guid":"44690160-29e2-4b31-8c98-68cb5f0d014a",
    "SystemFlag":false,
    "LdapFlag":false,
    "DomainId":null,
    "DistinguishedName":null,
    "DefaultHomeDashboardId":null,
    "DefaultHomeWorkspaceId":null,
    "UpdateInformation":
      {
      "CreateDate":"2016-05-26T13:41:11.677",
      "UpdateDate":"2016-05-26T13:41:11.677",
      "CreateLogin":2,
      "UpdateLogin":2
      }
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
  }
]
```

The following example shows the response if no group can be found.

```
[
  {
  "Links":[],
```

```
  "RequestedObject":{},
  "IsSuccessful":false,
  "ValidationMessages":
    [{
    "Reason": "WebApi:WebApiResourceNotFoundQueryReason",
    "Severity": 3,
    "MessageKey": "WebApi:WebApiResourceNotFoundQuery",
    "Description": "The resource cannot be found.",
    "Location": -1,
    "ErroredValue": null,
    "Validator":
    "ArcherApi.Controllers.System.UserGroupController, ArcherApi,
    Version=6.2.100.1061, Culture=neutral, PublicKeyToken=null",
    "XmlData": null,
    "ResourcedMessage": "No resource found."
    }]
  }
]
```

Remove group from role

Removes a group from an access role. Both group and access role must exist.

Request

```
PUT http://RsaArcher/api/core/system/rolegroup
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "GroupId":206,
  "RoleId":162,
  "IsAdd":false
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
```

```
    {
      "Id":206
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Remove group member

Removes a Group from another Group. GroupId is the ID of a parent group and GroupMemberId is the id of a child group. Both parent group and child group must exist.

Request

```
PUT http://RsaArcher/api/core/system/groupmember
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "GroupMemberId":208,
  "GroupId":162,
  "IsAdd":false
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":208
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Update group

Updates a group. The name and ID cannot be set to null.

The following table describes the results of including or excluding null values when updating a group.

Value	Result
ParentGroups null	Parent groups are not changed.
ParentGroups not null	Existing parent groups replaced by new list.
ChildGroups null	Child groups are not changed.
ChildGroups not null	Existing child groups replaced by new list.
ChildUsers null	Child users are not changed.
ChildUsers not null	Existing child users replaced by new list.

Request

```
PUT http://RsaArcher/api/core/system/group
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "Group":
    {
      "Id":208,
      "Name":"Updated Group Name",
      "DisplayName":"Updated Group Name",
      "Description":"<html><head></head><p style=\"margin:
      0px\">WAS: New Group Description</p></html>",
      "ParentGroups":null,
      "ChildGroups":null,
      "ChildUsers":null
    }
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":208
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Metadata Level

The Level and Level Layout segments are accessed through the System controller, which includes the following resources:

Get all levels

The Get all levels resource returns all levels in the current instance of the RSA Archer.

Request

```
POST http://RsaArcher/api/core/system/level
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get level by level ID

The Get level by level ID resource retrieves a level by the specified level ID.

Request

```
POST http://RsaArcher/api/core/system/level/*levelid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
```

```
Content-Type: application/json
X-Http-Method-Override: GET
```

Get levels by module ID

The Get levels by module ID resource retrieves the levels that are contained by the specified module ID.

Request

```
POST http://RsaArcher/api/core/system/level/module/*moduleid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get level by reference field ID

The Get level by reference field ID resource retrieves the levels targeted by the specified reference field ID. A cross-reference can target more than one level, which returns a collection. the target questionnaire cross-reference targets a single level, which is returned as a collection with one element.

Request

```
POST
http://RsaArcher/api/core/system/level/referencefield/*fieldid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get level layout by level ID

The Get level layout by level ID resource retrieves all levels in the current instance of RSA Archer.

Request

```
POST http://RsaArcher/api/core/system/levellayout/level/*levelid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Metadata Questionnaire

The Questionnaire segment is accessed through the System controller, which includes the following resources:

Get all questionnaires

The Get all questionnaires resource retrieves all questionnaires.

Request

```
POST http://RsaArcher/api/core/system/questionnaire
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get questionnaire by ID

The Get questionnaire by ID resource retrieves a questionnaire by the specified questionnaire ID.

Request

```
POST
http://RsaArcher/api/core/system/questionnaire/*questionnaireid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
```

```
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

The Questionnaire Rule segment is accessed through the System controller, which includes the following resources:

Get questionnaire rule by ID

The Get questionnaire rule by ID resource retrieves a questionnaire rule by the specified ID.

Request

```
POST
http://RsaArcher/api/core/system/questionnairerule/*questionnairer
uleid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get questionnaire rule by level

The Get questionnaire rule by level resource retrieves all questionnaire rules by the specified level.

Request

```
POST
http://RsaArcher/api/core/system/questionnairerule/level/*levelid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Metadata Role

The Role segment is accessed through the System controller, which includes the following resources:

Create role

The Create role resource creates an access role. The name field is required in the request body.

Request

```
POST http://RsaArcher/api/core/system/role
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
AcessRoleTasks(permssions) are supported in this phase
{"AccessRole": {"Name":"RoleA","Description":"This is a
testrole","IsDefault":false},"GroupIds":
[1,2],"AccessRoleTasks":null}
```

Delete role

The Delete role resource deletes an access role.

Request

```
DELETE http://RsaArcher/api/core/system/role/*roleid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Get all roles

The Get all roles resource retrieves all access roles.

Request

```
GET http://RsaArcher/api/core/system/role
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Get role memberships

The Get role memberships resource retrieves memberships for all roles. Membership includes all users and all groups for a role.

Request

```
GET http://RsaArcher/api/core/system/rolemembership
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Get role by user ID

The Get role by user ID resource retrieves the role for a user in the current RSA Archer instance.

Request

```
GET http://localhost/Archer/api/core/system/role/user/*userid*
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,applicatio
n/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Update role

The Update role resource updates an access role. The ID, alias, and name fields are required in the request body.

Request

```
PUT http://RsaArcher/api/core/system/role
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
AcessRoleTasks(permssions) are supported in this phase
{"AccessRole": {"Name":"RoleA","Description":"This is a
testrole","IsDefault":false},"GroupIds":
[1,2],"AccessRoleTasks":null}
```

Metadata Security

The Security Parameter segment is accessed through the System controller, which includes the following resources:

Create a security parameter

The Create a security parameter resource creates a security parameter.

Security parameter values

The following table shows the valid security parameter values when creating a security parameter.

Parameter	Valid Value
LockoutPeriodType	1 - (Hours), 2 - (Minutes), 3 - (Days)
SessionTimeoutType:	1 - (Hours), 2 - (Minutes), 3 - (Days)
DisallowedSessionDaysType:	1 - (Sunday), 2 - (Monday), 4 - (Tuesday), 8 - (Wednesday), 16 - (Thursday), 32 - (Friday), 64 - (Saturday)

Request

```
POST http://RsaArcher/api/core/system/securityparameter
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{"Name":"Test Security Param","Description":"test
sec","MinPasswordLength":9,"AlphaCharsRequired":2,"NumericCharsReq
uired":1,"UppercaseCharsRequired":1,"LowercaseCharsRequired":1,"Sp
ecialCharsRequired":1,"PasswordChangeInterval":90,"PasswordChangeL
imit":false,"GraceLogins":3,"MaximumFailedLoginAttempts":3,"Previo
usPasswordsDisallowed":10,"LockoutPeriod":999,"LockoutPeriodType":
3,"SessionTimeout":10,"SessionTimeoutType":2,"StaticSessionTimeout
":null,"PasswordExpirationNotice":30,"AutomaticAccountDeactivation
":0,"IsLimitByTimeFrame":false,"PermittedFromTime":"2015-03-
05T08:00:00","PermittedToTime":"2015-03-
05T10:00:00","IsDisallowedByDays":true,"DisallowedSessionDays":
[2,3],"IsDisallowedByDates":true,"DisallowedDates":
[{"LockedDate":"2015-04-01T00:00:00"},{"LockedDate":"2015-05-
01T00:00:00"}],"Default":false,"TimeZoneCode":"Central Standard
Time","SecurityParameterType":2}
```

Delete a security parameter

The Delete a security parameter resource deletes a security parameter.

Request

```
DELETE
http://RsaArcher/api/core/system/securityparameter/*securityparame
terid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Get all security parameters

The Get all security parameters resource retrieves all security parameters for the current RSA Archer instance.

Request

```
POST http://RsaArcher/api/core/system/securityparameter
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get security parameter by ID

The Get security parameter by ID resource retrieves a security parameter by the specified ID.

Request

```
POST
http://RsaArcher/api/core/system/securityparameter/*securityparame
terid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Update a security parameter

The Update a security parameter resource updates a security parameter.

Request

```
PUT http://RsaArcher/api/core/system/securityparameter
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body


```
{“Id”:23,”Alias”:”Test_Sec1”,”GUID”:”4B31AEEB-A85D-4665-B33F-
E3CC79021228”,"Name":"Test Security Param","Description":"test
sec","MinPasswordLength":9,"AlphaCharsRequired":2,"NumericCharsReq
uired":1,"UppercaseCharsRequired":1,"LowercaseCharsRequired":1,"Sp
ecialCharsRequired":1,"PasswordChangeInterval":90,"PasswordChangeL
imit":false,"GraceLogins":3,"MaximumFailedLoginAttempts":3,"Previo
usPasswordsDisallowed":10,"LockoutPeriod":999,"LockoutPeriodType":
3,"SessionTimeout":10,"SessionTimeoutType":2,"StaticSessionTimeout
":null,"PasswordExpirationNotice":30,"AutomaticAccountDeactivation
":0,"IsLimitByTimeFrame":false,"PermittedFromTime":"2015-03-
05T08:00:00","PermittedToTime":"2015-03-
05T10:00:00","IsDisallowedByDays":true,"DisallowedSessionDays":
[2,3],"IsDisallowedByDates":true,"DisallowedDates":
[{"LockedDate":"2015-04-01T00:00:00"},{"LockedDate":"2015-05-
01T00:00:00"}],"Default":false,"TimeZonCode":"Central Standard
Time","SecurityParameterType":2}
```


Metadata Selected User Group

The Selected User Group segment is accessed through the System controller, which includes the Get selected user group list by field ID resource.

Get selected user group list by field ID

This resource retrieves a list of SelectedUserGroups for the specified UserGroupListField by ID. This list represents the list of available selections of users and groups.

Request

```
POST
http://RsaArcher/api/core/system/usergroupselection/usergrouplist/
*fieldid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Metadata Subform

The Subform segment is accessed through the System controller, which includes the Get Subform by ID resource.

Get Subform by ID

This resource retrieves a Subform by the specified ID.

Request

```
POST http://RsaArcher/api/core/system/subform/*subformid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Metadata User

The User segment is accessed through the System controller, which includes the following resources:

Activate user

The Activate user resource changes the user status to Active.

Request

```
POST http://RsaArcher/api/core/system/user/status/active/*userid*
POST http://RsaArcher/api/core/system/user/status/active/1470
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
none
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":1470
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Add user to role

The Add user to role resource adds a user to an access role.

Note: The user and role must exist.

Request

```
PUT http://RsaArcher/api/core/system/userrole
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "UserId":1470,
  "RoleId":3,
  "IsAdd":true
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":3
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Add user to user group

The Add user to user group resource adds a user to a group.

Note: The user and group must exist.

Request

```
PUT http://RsaArcher/api/core/system/usergroup
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "UserId":1470,
  "GroupId":16,
  "IsAdd":true
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":16
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Change user password

The Change user password resource changes the user password.

Request

```
PUT http://RsaArcher/api/core/system/userpassword
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "UserId":1470,
  "NewPassword":"Brilliant123!"
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":{},
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Create user

The Create user resource creates a new user.

Rules for creating users

The new user name must be provided in the body of the request in the Content-Type header format.

When creating a user, adhere to the following rules:

- The following required fields must be completed:
  - FirstName
  - LastName
  - Password
- Valid AccountStatus are 1 (Active), 2(Inactive), 3(Locked).
- If Roles are not provided, the default is General User Role.
- If Language ID is not provided, the default language is used.
- The default user name is LastnameFirstnameinitial IE: doej for John Doe.

If the save is successful, the requested object returned from the URI includes the ID of the user. If the save is not successful, validation messages explaining the nature of the failure are returned.

Request

```
POST http://RsaArcher/api/core/system/user
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "User":
    {
      "FirstName":"John",
      "LastName":"Doe"
    },
  "Password":"NewUser2005!"
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":1470
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Deactivate user

The Deactivate user resource changes the user status to Inactive (value is 2).

Request

```
POST
http://RsaArcher/api/core/system/user/status/inactive/*userid*
POST http://RsaArcher/api/core/system/user/status/inactive/1470
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
```

```
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
none
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":1470
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Delete user

The Delete user resource deletes a user.

Request

```
DELETE http://RsaArcher/api/core/system/user/*userid*
DELETE http://RsaArcher/api/core/system/user/1471
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
none
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
    {
      "Id":1470
    },
  "IsSuccessful":true,
```

```
  "ValidationMessages":null
}
```

Get all users

The Get all users resource retrieves all users in the current RSA Archer instance.

Request

```
POST http://RsaArcher/api/core/system/user
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Request Body

```
none
```

Response Example

Note: This example is one user. The command retrieves as many users as there are in your environment.

```
[
  {
  "Links":[],
  "RequestedObject":
    {
    "Id":229,
    "DisplayName":"Archer, Eric",
    "FirstName":"Eric",
    "MiddleName":"",
    "LastName":"Archer",
    "LastLoginDate":"2016-07-28T17:19:01.137",
    "UserName":"ericsc",
    "AccountStatus":1,
    "DomainId":null,
    "SecurityId":6,
    "Locale":"en-US",
    "TimeZoneId":"Eastern Standard Time",
    "Address":"",
    "Company":"",
```

```
    "Title":"",
    "AdditionalNote":null,
    "BusinessUnit":null,
    "Department":null,
    "ForcePasswordChange":false,
    "DistinguishedName":null,
    "Type":1,
    "LanguageId":null,
    "DefaultHomeDashboardId":-1,
    "DefaultHomeWorkspaceId":-1,
    "UpdateInformation":
      {
      "CreateDate":"2015-06-29T17:12:29.107",
      "UpdateDate":"2016-07-18T20:01:12.333",
      "CreateLogin":2,
      "UpdateLogin":229
      }
    },
    "IsSuccessful":true,
    "ValidationMessages":[]
  }
]
```


Get all user contacts

The Get all user contacts resource retrieves contact information for all users in the current RSA Archer instance.

Request

```
GET http://localhost/Archer/api/core/system/usercontact
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
none
```

Response Example

Note: This example is one contact for a user. The command retrieves as many contacts for the user as there are in your environment.

```
[
  {
  "Links":[],
  "RequestedObject":
    {
    "UserId":1470,
    "Contacts"
      [
        {
        "ContactType":7,
        "ContactSubType":2,
        "IsDefault":true,
        "Value":"example@domain.com",
        "Id":1135
        }
      ]
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
  }
]
```

Get contact information for a user

The Get contact information for a user resource retrieves contact information for a user in the current RSA Archer instance.

Contact types and contact subtypes

When the Contact Type is set to 9 - Phone, Contact Sub Types 3 - 14 are available. When the Contact Type is set to 7 - Email, Contact Sub Types 1, 4, 7,8, and 13 are not available.

The following table shows the numbers and definitions of contact types and sub-types.

Contact Type	Definition	Contact Sub Type	Definition
7	Email	1	Assistant
9	Phone	2	Business

Contact Type	Definition	Contact Sub Type	Definition
		3	Business2
		4	Business Fax
		5	Home
		6	Home2
		7	Home Fax
		8	ISDN
		9	Mobile
		10	Mobile2
		11	Other
		12	Other2
		13	Other Fax
		14	Pager

Request

```
GET http://localhost/Archer/api/core/system/usercontact/*userid*
GET http://localhost/Archer/api/core/system/usercontact/1470
```

Request Header

```
Accept:application/json,text/html,application/xhtml+xml,applicatio
n/xml;q=.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
none
```

Response Example

```
[
  {
  "Links":[],
  "RequestedObject":
```

```
    {
    "ContactType":7,
    "ContactSubType":2,
    "IsDefault":true,
    "Value":"example@domain.com",
    "Id":1135
    },
  "IsSuccessful":true,
  "ValidationMessages":[]
  }
]
```

Get user by ID

The Get user by ID resource retrieves a user by the specified ID.

Request

```
POST http://RsaArcher/api/core/system/user/*userid*
POST http://RsaArcher/api/core/system/user/1470
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Request Body

```
none
```

Response Example

```
{
"Links":[],
"RequestedObject":
  {
  "Id":1470,
  "DisplayName":"Doe, John",
  "FirstName":"John",
  "MiddleName":null,
  "LastName":"Doe",
  "LastLoginDate":"2016-09-13T15:16:18.35",
  "UserName":"doej",
  "AccountStatus":1,
```

```
  "DomainId":null,
  "SecurityId":1,
  "Locale":null,
  "TimeZoneId":"Eastern Standard Time",
  "Address":null,
  "Company":null,
  "Title":null,
  "AdditionalNote":null,
  "BusinessUnit":null,
  "Department":null,
  "ForcePasswordChange":false,
  "DistinguishedName":null,
  "Type":1,
  "LanguageId":null,
  "DefaultHomeDashboardId":-1,
  "DefaultHomeWorkspaceId":-1,
  "UpdateInformation":
    {
    "CreateDate":"2016-09-12T19:30:49.043",
    "UpdateDate":"2016-09-13T17:54:48.807",
    "CreateLogin":2,
    "UpdateLogin":2
    }
  },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```


Get users by group

The Get users by group resource retrieves all users that are members of the specified group.

Request

```
POST http://RsaArcher/api/core/system/user/group/*groupid*
POST http://RsaArcher/api/core/system/user/group/85
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Request Body

```
none
```

Response Example

Note: This example is just one user. This command retrieves as many users as there are in the given group.


```
[
  {
  "Links":[],
  "RequestedObject":
    {
    "Id":1470,
    "DisplayName":"Doe, John",
    "FirstName":"John",
    "MiddleName":null,
    "LastName":"Doe",
    "LastLoginDate":"2016-09-13T15:16:18.35",
    "UserName":"doej",
    "AccountStatus":1,
    "DomainId":null,
    "SecurityId":1,
    "Locale":null,
    "TimeZoneId":"Eastern Standard Time",
    "Address":null,
    "Company":null,
    "Title":null,
    "AdditionalNote":null,
    "BusinessUnit":null,
    "Department":null,
    "ForcePasswordChange":false,
    "DistinguishedName":null,
    "Type":1,
    "LanguageId":null,
    "DefaultHomeDashboardId":-1,
    "DefaultHomeWorkspaceId":-1,
    "UpdateInformation":
      {
      "CreateDate":"2016-09-12T19:30:49.043",
      "UpdateDate":"2016-09-13T17:54:48.807",
      "CreateLogin":2,
      "UpdateLogin":2
      }
    },
```

```
  "IsSuccessful":true,
  "ValidationMessages":[]
  }
]
```

Remove user from user group

The Remove user from user group resource removes a user from a group.

Note: The user and group must exist.

Request

```
PUT http://RsaArcher/api/core/system/usergroup
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "UserId":1470,
  "GroupId":85,
  "IsAdd":false
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":{"Id":85},
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Remove user from role

The Remove user from role resource removes a user from an access role.

Note: The user a role must exist.

Request

```
PUT http://RsaArcher/api/core/system/userrole
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "UserId":1470,
  "RoleID":25,
  "IsAdd":false
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":{"Id":35},
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Update user

The Update user resource updates an existing user. When updating a Domain user, two additional properties are added: SecurityID and DomainID. If DomainID is not given, the user account Domain shows No Domain, and if SecurityID is not given, the user account is set to the default parameter.

Note: The FirstName, LastName, UserName, Id, and AccountStatus properties cannot be null.

Null in user values

The following table describes the results of including or excluding null values when updating a user.

Value	Result
Contacts list null	User contacts are not changed.
Contacts list not null	User contacts are replaced by new list.
Roles list null	User roles are not changed.
Roles list not null	User roles are replaced by new list.
Groups list null	User groups are not changed.

Value	Result
Groups list not null	User groups are replaced by new list.

Request

```
PUT http://RsaArcher/api/core/system/user
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
```

Request Body

```
{
  "User":
  {
    "ID":"1470",
    "FirstName":"John",
    "LastName":"Doe",
    "UserName":"DoeJ",
    "AccountStatus":"1"
  },
  "Contacts":
  [
    {
      "ContactType":7,
      "ContactSubType":2,
      "Value":"none@none.com",
      "IsDefault":true
    },
    {
      "ContactType":9,
      "ContactSubType":2,
      "Value":"9999999999"
    }
  ],
  "Roles":[1],
  "Groups":[]
}
```

Response Example

```
{
  "Links":[],
  "RequestedObject":
  {
    "Id":1470
  },
  "IsSuccessful":true,
  "ValidationMessages":[]
}
```

Metadata Values List Values

The Values List Definitions and Values List Values segments are accessed through the System controller, which includes the following resources:

Get values list definition by ID

The Get values list definition by ID resource retrieves the definition of a values list by its ID.

Request

```
POST http://RsaArcher/api/core/system/valueslist/*valueslistid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get values list values by values list ID (flat)

The Get values list values by values list ID (flat) resource retrieves a flat list of all ValuesListValues for the specified ValuesListDefinition.

Note: This list is in a flat format. To get the list in a tree format, use the following URI: http://RsaArcher/api/core/system/valueslistvalue/valueslist/*valueslistid*

Request

```
POST
http://RsaArcher/api/core/system/valueslistvalue/flat/valueslist/v
alueslistid
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```

Get values list values by values list ID (hierarchical)

The Get values list values by values list ID (hierarchical) resource retrieves a hierarchical list of all ValuesListValues for the specified ValuesListDefinition.

Note: This list is in a tree format. To get the list in flat format, use the following URI: http://RsaArcher/api/core/system/valueslistvalue/flat/valueslist/*valueslistid*

Request

```
POST
http://RsaArcher/api/core/system/valueslistvalue/valueslist/*value
slistid*
```

Request Header

```
Accept:
application/json,text/html,application/xhtml+xml,application/xml;q
=0.9,*/*;q=0.8
Authorization: Archer session-id="*SessionToken"
Content-Type: application/json
X-Http-Method-Override: GET
```