AMCS PLATFORM
REST API User Guide

Revision History

Revision	Date	Revised By	Notes
8.5a	23/03/2020	Ric Morris	User guide for version 8.5 of Platform
8.5b	22/04/2020	Ric Morris	Information on PAT token authorisation added.

Contents

1 Overview

This document describes the REST API of ELEMOS version 8.3 and above.

1.1 Who Should Read this Document

This document contains a technical description of the AMCS Platform REST API. It is intended for:

- System integrators
- Customer IT team looking to integrate AMCS Platform with another system
- Third parties looking to integrate their system with AMCS Platform

1.2 Document Scope

This document introduces the technical aspects of the AMCS Platform REST API. The purpose of this document is to provide readers with detail on how to:

- Access the REST API
- Make calls to the API to retrieve, append and update resources
- Retrieve the incremental changes to a collection
- Find more detail on the end points that are available from an environment.

1.3 Summary

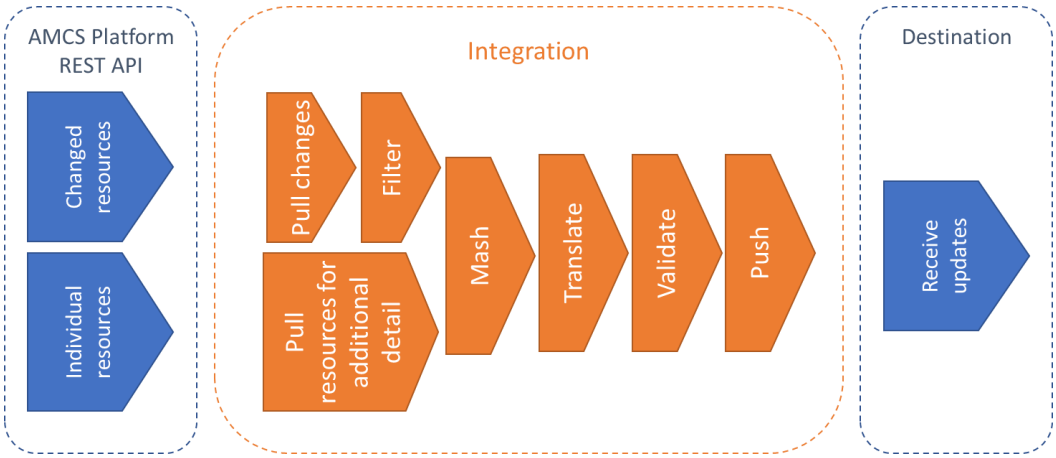
AMCS Platform REST API is designed to support the integration of AMCS Platform with other systems. With care and additional infrastructure, the REST API can be used to build functionality on top of Platform but that is not it's primary purpose.

The 'changes since' process provides the main means of performing an initial synchronisation of the AMCS Platform and another system, and then keeping them synchronised. Methods are also provided to create and update resources.

2 Integration Patterns and Tools

The AMCS Platform provides a number of tools that can be used to connect systems. Care should be taken to match the right tooling to the intended pattern of integration.

- When an extract-based approach is needed, use BI reporting.
- When derived values are require, use the data mart.
- When business processes on different systems need to be integrated by synchronising the data on each through an event style interface, use the REST API.



An integration using the REST API will need to take full responsibility for all aspects of the integration including:

- Pulling change events from AMCS Platform
- Filtering the changed resources for the ones that are of interest
- Pulling additional resources from the REST API in order the retrieve all the properties needed by the destination
- Mashing resources to create the complete view required
- Translating that into the form required by the destination
- Validating this to ensure it meet the entry rules of the destination
- Pushing the result to the destination

In the early stages of designing the interface care should be taken to ensure that the need for caching/staging has been considered. The changes since functionality is effectively a means of pulling create/change events from platform. The only filters available enable retrieval of related resources to build a view external to AMCS Platform.

3 Basics

The AMCS REST API provides a RESTful interface that allows access to the primary business resources of the AMCS platform. The interface follows general HATEOAS compliance principles.

3.1 Interface definitions

Swagger documentation for the end points available on an environment can be found at:

Before 8.5: https://{environment domain name}/swagger/ui/index

From 8.5: https://{environment domain name}/erp/api/swagger/ui/index

3.2 HTTP Interface

The following HTTP RESTful methods are provided as part of this API:

- GET: Provides read access to retrieve details of one or more resources.
- POST: Creates or updates a single resource or a set of resources.
- PUT: Updates a single resource or a set of resources.

Note:

- DELETE methods are currently not provided as part of the API.

The REST API URIs begin with

https://{environment domain name}/erp/api/integrator/erp

For example:
https://prod.amcs-at-mycompany.com/erp/api/integrator/erp

This should be used as the base of the URI. For instance, the full URI for the partial URI /directory/customers would be:

https://prod.amcs-at-mycompany.com
/erp/api/integrator/erp/directory/customers

Versioning

The x-api-version HTTP header should be used to indicate a specific version of an endpoint. e.g. x-api-version:8.3 will make a request to the version of endpoint released with 8.3 of the AMCS Platform.

A request without the x-api-version HTTP header will default to the latest version of the endpoint available.

Content type and encoding

All requests should contain the correct content type. Content can be requested and passed in any encoding that can be mapped to utf-16. The AMCS REST API requires the following headers where applicable (i.e.: sending a JSON payload).

- Content-Type:application/json
- Accept:application/json

Result codes

Appendix A contains a comprehensive list of possible HTTP status codes.

3.3 Naming Conventions

The AMCS REST API follows JSON and mixed Camel and Pascal notation on the format of the fields and entities returned.

Related properties

Properties that link a resource with another resource begin with "related". Most "related" properties have the GUID of the related resource as a value. i.e.: relatedCustomerGuid in a /sites resource endpoint contains the Guid for the customer that site belongs to.

Where more than one resource may be related the "related" property will consist of a list. This list may contain a simple list of GUIDs or it may contain a list of objects, with each object giving the detail of a relationship. The properties of these objects are the properties of the relationship, rather than the properties of the related resource.

ListItem properties

Properties that are based on a selection from a list are named with "ListItem" at the end. The possible options for these properties are static or change via configuration. The list of possible values can be found by making a GET request to /lists/{property name with ListItem removed and pluralised}.

The response contains a JSON object containing an array of objects, each containing the GUID and details for each possible value.

GET /integrator/lists/{property name with Enum removed and pluralized }/{guid} is also supported if you only want the details of an individual property.

Standard-compliant properties

Properties with the following ending must comply with the indicated standard:

Type	Standard
Dates	RFC3339 subset of ISO8601
Identifiers	RFC4122

4 Methods

This chapter specifies the different methods to be used when using the AMCS REST API.

4.1 Retrieving resources

Retrieving a resource or list of resources is done with the GET request. The request could be sent to the resource collection or the specific GUID of a resource.

/{grouping}/{collection} - requests the properties for all the resources in a collection.
/{grouping}/{collection}/{guid} - requests the properties for a single resource

The result of either request is an array of resources for instance:

GET /accounting/payments

Response:

```
{
    "resource": [
        {
            "resource": {
                "Name": "Test",
                "Reference": null,
                "ARAccountCode": null,
                "FederalId": null,
                "IsInternal": false,
        ...
       }
    ]
}
```

GET /payments/12345678-1234-1234-12345678ab

Response:

```
{
    "resource": {
        "Name": "Hany",
        "Reference": null,
        "ARAccountCode": null,
        "FederalId": null,
        "IsInternal": false,
        "ReceiveServiceUpdatesByEmail": false,
     ...
   }
}
```

Pagination

A GET to a collection results in a paginated response. To select the page use the "page" and "max" query parameters. For instance, the following request will return 100 items for page 3.

/{grouping}/{collection}?page=3&max=100

Filters

To receive a filtered set of resources from a collection, the API offers as part of the GET request to /{collection} a filter parameter that can be specified with a selector property.

For instance: /{collection}?filter={property} eq {value}

The properties that can be filtered are called out in the Swagger documentation. These are typically the properties starting “Related” plus the references that allow resources to be unequally identified.

4.2 Retrieving changed resources

The AMCS REST API provides a way to retrieve changes that have been made since a point in the collection history by using the following URI /{collection}/changes?since={bookmark}

The API will provide as response a resource array containing the resources that have been created or updated since the bookmark.

As part of the response a "next" URI is provided. This URI that can be called to get the next batch of resources. A new bookmark is also provided for use in one of two ways:

When there are more changes available to be retrieved the bookmark is provided via a "cursor" property. A new cursor bookmark is provided with each response. The URI needs updating with the new bookmark each time, or just call the "next" URI that is provided with each response.

Use the following URI to get the next batch of new or updated resources:

/{collection}/changes?cursor={bookmark}

When the response contains all the changes available the bookmark is provided via an "until" property. The next time you are ready to check for resources use this in a URI as follows. Cursor bookmarks cannot be used in a "changes?since" URI. Until bookmarks cannot be used in a "changes?cursor" URI.

Use the following URI to get the resources changed since:

/{collection}/changes?since={bookmark}

Notes:

- The number of resources in the response is limited in order to maintain the overall performance of the AMCS Platform.
- It is expected that API clients make more than one call to retrieve all the changes that have been made to the collection.

For example:

/customers/changes?since=890099ab99c0f

Response:

```
{
    "resource":[
        ...
    ],
    "links":{
        "next":"#/customers/changes?cursor='absle3290'"
    },
    "extra":{
        "cursor":"absle3290"
    }
}
```

4.3 How to create resources

The AMCS API Platform will create a resource when the client sends a POST request to a /{grouping}/{collection} without a GUID. The body of the request should contain a single JSON object containing all the properties of the resource. The value of Enum properties should an object containing a "Guid" property with the GUID of the value required.

For example, to create a customer:

POST /customers

```
{
    "Name":"My Customer",
    "CustomerTypeEnum":{
        "guid":"12345678-1234-1234-1234-12345678ab"
    },
    ...
}
```

Example 1: Create a resource

4.4 How to update resources

The AMCS Platform API will create a resource when the client sends a POST request to a /{collection} without a GUID. The body of the request should contain a single JSON object containing all the properties of the resource. The value of Enum properties should an object containing a "Guid" property with the GUID of the value required.

Notes:

- A POST request with a GUID to a collection replaces the entire resource. Any properties that are not supplied will result in their value being lost (set to NULL).

- To update part of a resource without impacting the other properties a PUT request to /{collection}/{resource guid} should be used. The request body should contain a JSON object containing the properties to be updated.

For example, to update a customer:

POST /customers

```
{
    "GUID":"12345678-1234-1234-1234-12345678ab",
    "Name":"My Customer",
    "CustomerTypeEnum":{
        "guid":"12345678-1234-1234-1234-12345678ab"
    },
    ...
}
```

5 Security

The API uses HTTPS as transport layer for security and uses HTTP/1.1 – RFC 7230 for message syntax and routing.

There are 2 ways of authenticating in order to access the REST API.

- Requesting a bearer token from /authTokens using username and password
- Requesting a session log on from /authTokens using a PAT token

Authentication using username and password will be phased out after 8.6.

5.1 GET /authTokens with username and password direct to the App server

Before using this method a user must be set up with access right "Can POST, PUT, GET to the REST API". This can be done from User Management in the AMCS Platform UI.

Request format

The following table specifies the format and parameters for this request.

Method	Type	Name	Type
GET	Parameter	Username	URI encoded string
	Parameter	password	URI encoded string

Response format

The following table specifies the format and parameters for the response.

Successful response	Type
Bearer token	text/html

The token returned is then uses as a Bearer token in the subsequent calls to the REST API

Example

The following shows an example of a request to the authentication endpoint:

https://{elemos-url}/authTokens?username=john&password=password

The username and password in this request may need to be URI encoded.

The token in the response should be supplied in all subsequent requests in an authorization HTTP header.

Authorization: Bearer tcpRoNbIKOJZj9Yn0CL2oiX7m04JJmlAIlaskd810incqjslasadjzxzza29axb

Notes:

- No specific Accept or Content-Type header is required for the authentication endpoint as it will always return text/html as content.

5.2 POST /erp/api/authTokens with PAT token via the core App

Before using this method, a user must be set up with:

- Access right “Can POST, PUT, GET to the REST API”
- A PAT token

This can be done from User Management in the AMCS Platform UI.

Request format

The following table specifies the format and parameters for this request.

Method	Body
POST	{ “privatekey”:”{PAT token}” }

Response format

The response passes a Platform.ERP.SessionToken as a cookie.

This cookie should be included in the subsequent calls to the REST API.

6 Appendix A

The following table shows the cause codes that could be returned as a result of a request to the AMCS REST API.

Header	Value
200	OK – All worked as expected
400	Bad request – The request couldn't be accepted. This is normally due an invalid or missing required parameter.
401	Unauthorized – No valid username/password, API key or Bearer token provided.
402	Request failed – The request was formatted correctly but failed due some inconsistent parameters.
403	Forbidden – The username/password, API key or Bearer token provided does not have permissions to access the resource requested.
404	Not Found – The URI or resource was not found or does not exist.
409	Conflict – The request conflicts with an existing request or resource. This is normally due to duplicated reference keys or idempotent keys.
429	Too many requests – The request couldn't be service due do too many requests service too quickly on the same endpoint. A back-off of the request rate is recommended.
500 or 50X (range)	Something went wrong at the server side, please try again.