ARIS REPOSITORY API - TECHNICAL INTRODUCTION

ARIS 10

October 2026

This document applies to ARIS Version 10 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2010-2026 SAG Aris GmbH, Saarbrücken, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name ARIS and all SAG Aris GmbH product names are either trademarks or registered trademarks of SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

The Software may contain or be distributed with certain commercial third party components (“Third Party Components”), which may be subject to special license terms and conditions (“Third Party Terms”) located at https://aris.com/terms.

Third Party Terms include important licensing and warranty information and disclaimers of third party licensors. For the avoidance of doubt, if Customer uses the Software in accordance with the Documentation, the Third Party Terms do not restrict the rights granted under this Agreement. In the event that a third party manufacturer no longer provides active support for any Third Party Component, Supplier will use commercially reasonable efforts to respond to any issues with the Software related to the Third Party Components.

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors.

Table of Contents

1. Introduction 4
2. General principles 5
3. Login/Logout 6
    3.1. Obtaining a UMC session token 6
    3.2. Releasing a UMC session token 8
4. Get information about ARIS Method 9
5. Get information on available databases 11
6. API Docs ComparePair 13
    6.1. Get a list of database comparison pairs 13
    6.2. Create a database comparison pair 14
    6.3. Toggle the enable state of a database comparison pair 15
    6.4. Delete a database comparison pair 16
7. Get an item 18
8. Get group children 20
9. Get content of a model 21
10. Create an item 23
11. Create model content 25
12. Delete model content 27
13. Move an item 28
14. Attributes 29
    14.1. Retrieval 29
    14.2. Creation 30
    14.3. Updating or creation for existing item 30
    14.4. Deleting an attribute 31
    14.5. Styled values for text attributes 32
15. Assignments 34
16. Variants 35
17. Model graphic 36
18. Finding items in the database 37
19. Paging 40
20. Ordering the result 42
21. Generic queries 43
    21.1. Use case: Get connected objects (def level) 45
    21.2. Use case: Get connected objects (occ level) 46
    21.3. Use case: Get models with occurrences of an object 48
    21.4. Use case: Get objects to which a model is assigned 49
    21.5. Use Case: Get models with occurrence pattern 49
    21.6. Use Case: Get models with occurrence pattern 50
22. Support and legal information 52
    22.1. Documentation scope 52
    22.2. Support 52
    22.3. Legal notices 53

1. Introduction

This document provides additional information for accessing an ARIS repository via the ARIS REST API. It should be considered as a supplement to the official API documentation available on each running ARIS Server and ARIS Design Server under **https://<servername:port>/apidocs**. If you do not specify a port number and use **https://<servername>/apidocs** instead, the default port is used, for example, 80.

2. General principles

- The ARIS RESTful APIs are designed to access an ARIS repository by apps typically running on mobile devices. It is not meant as a replacement for other ways to access an ARIS repository, for example, the ARIS Report API. Therefore, it has various limitations with respect to functionality that you might have expected.
- Every call to the API is atomic: either the operation succeeds or fails.
- Every call to the API requires a valid API cookie/token from ARIS User Management.
- Many calls have obligatory and/or optional URL parameters. All parameter values must be URL-encoded as they may contain special characters.
- If URL parameters are passed that are unknown or not the correct ones for the specific method or misspelled, then they are silently ignored. This may lead to an unexpected outcome of the requested operation.
- Many calls require a database language as parameter. If not given, the fallback language of the current database is used.
- Many operations require an ARIS Method filter GUID as parameter. If not given, a specific auto-selection mechanism chooses the right filter, like to a login on the ARIS portal. It is strongly recommended to pass the desired filter as parameter which is also much faster.
- The result objects may contain method data, for example, typename. Method data is delivered in the language from the client's HTTP header ("accept-language"). Alternatively, you can pass an optional URL parameter methodlanguage in order to set the method language directly.
- For read-requests, there is an absolute result size limit. The limit is typically 100,000 for object items and 10,000 for all other items. like models or groups (different limits may be applicable depending on the user license or server settings).
- This means that the sum of all result items from all paging requests cannot exceed the absolute result size limit [40].
- For write-requests, for example the creation of objects, the limit is 500 items per request
- Date parameters must be provided and are returned in UTC in the RFC 3339 Internet format YYYY-MM-DD, and for timestamps in UTC in the RFC 3339 Internet Zulu time format is needed YYYY-MM-DD'T'HH:MM:SS'Z', for example, 2023-01-22T08:22:55Z.

3. Login/Logout

For all API calls, a valid token from ARIS User Management (also known as UMC) is required. The token must be obtained by calling a dedicated method from the UMC API. It is not allowed to use a standard UMC token obtained by logging in to the Connect portal.

The token must be included in all subsequent API calls as URL parameter (umcsession) or added as cookie to the HTTP request.

3.1. Obtaining a UMC session token

The endpoint for obtaining the UMC token is:

POST

http(s)://<servername:port>/umc/api/v2/tokens

Required URL parameters are:

- tenant
- name
- password

Basic authentication is also supported (see remarks below).

Example

POST

```
https://<servername:port>/umc/api/v2/tokens?
tenant=default&name=system&password=manager
```

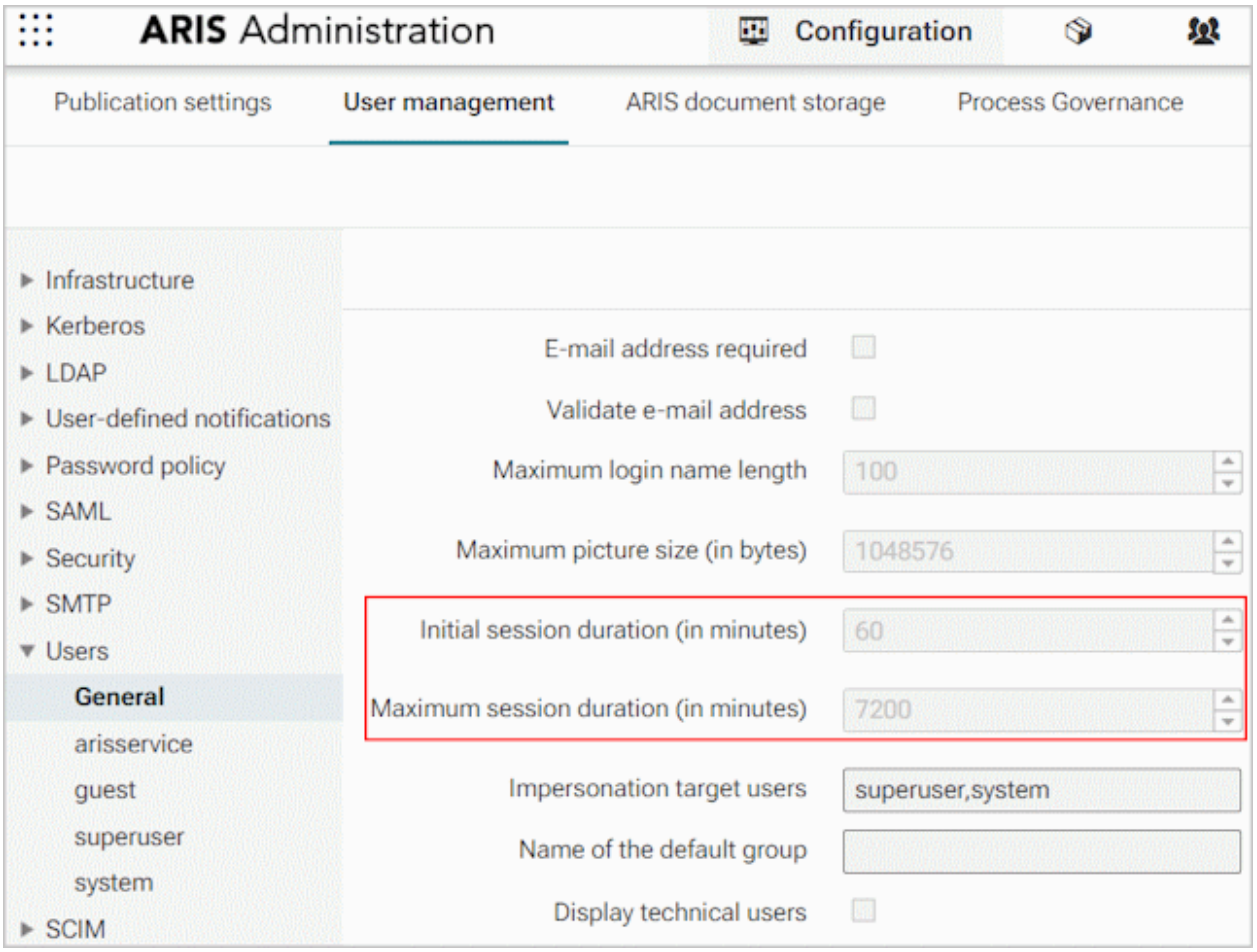
Response:

```
{
  "csrfToken": "kZcIkOV2xrD3u3M5bKy9NvzY_TriwE1KmC6k6huUWxQ",
  "token" : "eyJhbGciOiJIUzI1NisImp0aSI6FjY "
}
```

Remarks

- All URL parameters must be urlencoded if they contain special characters like umlauts, non-ASCII, etc.
- A successful invocation returns a UMC cookie in the HTTP session and in its JSON response body the UMC token as string.

- For all subsequent API requests you can either provide the cookie orhand over the token as URL parameter **umcsession** (do not forget to urlencode the token) or pass the token as bearer token in the HTTP header: Authorization: Bearer <yourtoken>
- The JSON response of the request to /umc/api/v2/tokens also contains a CSRF-token. This CSRF-token must be passed as URL parameter **csrfToken** (urlencoded!) in subsequent requests if the REST API is called from a web browser environment (or more precisely stated: if the user-agent provided by the calling client indicates a browser like Chrome or Mozilla Firefox®).
- As many subsequent API calls as desired can be carried out with the same UMC token/cookie as long as the UMC maximum session duration limits are not exceeded. These limits can be viewed/edited in the User management of the ARIS Administration:



- When no more API requests will be sent for some time it is strongly recommended to explicitly "logout" again in order to release the consumed ARIS license (see next chapter:).

  Releasing a UMC session token
  When the UMC session is no longer required, it should be explicitly invalidated, thus releasing the consumed ARIS license. The endpoint to use is:

```
DELETE https://<servername:port>/umc/api/tokens/<token>
```

Example

DELETE

```
https://<servername:port>/umc/api/tokens/eyJhbGciOiJIUzI1NisImp0aSI6FjY
```

- It is also possible to obtain the token with preemptive basic authentication. In this case, user name and password are not specified as parameters. Instead, the HTTP header must contain the entry **Authorization: Basic xyz,** where **xyz** is the BASE64 representation of the string **name:password**.

3.2. Releasing a UMC session token

When the UMC session is no longer required, it should be explicitly invalidated, thus releasing the consumed ARIS license. The endpoint to use is:

```
DELETE https://<servername:port>/umc/api/tokens/<token>
```

Example

DELETE

```
https://<servername:port>/umc/api/tokens/eyJhbGciOiJIUzI1NisImp0aSI6FjY
```

4. Get information about ARIS Method

```
GET https://<servername:port>/abs/api/methodology/modeltypes
```

This call returns a list with all available model types.

```
GET https://<servername:port>/abs/api/methodology/objecttypes
```

This call returns a list with all available object types.

```
GET https://<servername:port>/abs/api/methodology/cxnbasetypes
```

This call returns a list with all available connection base types.

```
GET https://<servername:port>/abs/api/methodology/symboltypes
```

This call returns a list with all available symbol types.

```
GET https://<servername:port>/abs/api/methodology/attributetypes
```

This call returns a list with all available attribute types.

```
GET https://<servername:port>/abs/api/methodology/attributevaluetypes
```

This call returns a list with all available attribute value types.

The calls support four optional URL parameters: **pagesize** and **pagetoken** to control the paging (see Paging [40]), **methodfilter** to retrieve the types which are allowed in this filter, **methodlanguage** to specify the locale of the method types (see also General principles [5]).

Example with method filter and language

GET https://<servername:port>/abs/api/methodology/modeltypes?methodfilter=8e386410-e0e0-11de-242a-f28d40dfaca6&methodlanguage=en_US

The result is a list with the method types. For each method type is returned the name, type number, API name and GUID.

Example for a model type


```
{
      "kind": "METHODTYPE",
      "name": "BPMN allocation diagram (BPMN 2.0)",
      "type": 252,
      "apiname": "MT_BPMN_ALLOCATION_DIAGRAM",
      "type_guid": ""
}
```

Example for an object type


```
{
	"kind": "METHODTYPE",
	"name": "Function",
	"type": 22,
	"apiname": "OT_FUNC",
	"type_guid": ""
}
```

5. Get information on available databases

GET

```
https://<servername:port>/abs/api/databases?language=en&attributes=all
```

This call returns a list of all available databases for the current user. It is the same list that a user would see when opening ARIS Architect.

For each database, its name, main group GUID, and some interesting flags are returned. It is also possible to retrieve the maintained attributes by using the URL parameter **attributes**. You can find more about attributes in section **Retrieval [29]**.

Example for Database item

```
{
"kind": "DATABASE",
"name": "Sample DB",
"isversioned": true,
"maingroup_guid": "4a713de0-5d02-11e3-0fda-fd81e986d7e2"
}
```

Note: A license of type **ARIS API Full Access**, **ARIS API System Access**, **ARIS API Read Access** or a sufficient ARIS product license is required for this operation.

GET

```
https://<servername:port>/abs/api/databases/Sample%20DB?
language=en&attributes=all
```

This call returns more detailed information for the given database name: main-group GUID, **isversioned flag** and **ispublished** flag, all maintained attributes in database language EN as well as a list of all allowed method filters and database languages. Exactly one database language will have the flag **isalternative = true** which means that it will be used as the fallback database language if a request does not pass the URL parameter **language**.

In case of a versionable database, the existing change lists are returned as well.

Example for method filter item

```
{
"kind": "METHODFILTER",
"guid": "dd838074-ac29-11d4-85b8-00005a4053ff",
"name": "Entire method",
"description": "All method content is available."
}
```

Example for database language item

```
{
"kind": "DBLANGUAGE",
"language": "en_US",
"isalternative": true
}
```

Example for changelist item

```
{
"kind": "CHANGELIST",
"changelist_number": 1,
"user": "internal",
"description": "initial revision after restore from basic archive file.",
"submit_time": "2015-04-07T16:11:13Z"
}
```

6. API Docs ComparePair

Note: The ComparePair feature is not part of the ARIS Repository REST API but is related to CONNECT Portal Functionality. It is included here for convenience.

6.1. Get a list of database comparison pairs

GET

```
/publishing/databaseComparePair
```

https://<servername:port>/abs/publishing/databaseComparePair

This call returns a list of all available database pairs.

Example

```
[
        {
        "name": "A",
        "user": null,
        "description": null,
        "sourceDatabaseName": "test_mv",
        "sourceVersion": 1,
        "targetDatabaseName": "test_mv",
        "targetVersion": 2,
        "enabled": true
        }
]
```

GET

/api/databaseComparePair(dbPairName)

https://<servername:port>/abs/publishing/databaseComparePair/A

Parameter

```
dbPairName = A      // Name of the database pair
```

Gets a database pair for comparison by its name.

This call returns a single database pair if one is found.

The information detail is exactly the same as in **get databaseComparePair**.

No new information is provided.

Example


```
{

  "kind": "RESULT",
"request": "abs#getDatabaseComparePair",
"status": "OK",
"item_count": 1,
"items": [ {
       "kind": "DATABASE",
       "name": "A",
       "user": null,
       "description": null,
       "sourceName": "test_mv",
       "sourceVersion": 1,
       "targetName": "test_mv",
       "targetVersion": 2,
       "enabled": false
} ]
}
```


6.2. Create a database comparison pair

POST

/api/databaseComparePair

https://<servername:port>/databaseComparePair?
name=A&user=test&description=test%20description&sourceDatabaseName=Sample%20DB&sourceVersion
=1&targetName=Sample%20DB&targetVersion=2

Parameter

```
databasePairName             = A          //Name of the database pair
user                         = ""         //Name of the responsible user
description                  = ""         //Description of the pair
sourceDatabaseName           = Test_mv    //Name of the source database
sourceVersion                = 1          //Version of the source database
targetDatabaseName           = Test_mv    //Name of the target database
targetVersion                = 2          //Version of the target database
enabled                      = true      //Enable the database pair
```

This call returns a list of a newly created database pair.

For each pair a name, a short description, and the responsible user can be saved.

The unique name of the database pair is required.

The source database must be unique. The target database can be referenced in several pairs.

For source and target, the database and version must exist, but the databases do not need to be published.

The versions do not have to be different either, but if the source and the target database are the same, they can be exchanged.

Database pairs with the same name or the same source database overwrite older ones.

Example


```
{

  "kind": "RESULT",
"request": "abs#createDatabaseComparePair",
"status": "OK",
"item_count": 1,
"items": [ {
      "kind": "DATABASE",
      "name": "A",
      "user": "",
      "description": "",
      "sourceName": "test_mv",
      "sourceVersion": 1,
      "targetName": "test_mv",
      "targetVersion": 2,
      "enabled": true
} ]
}
```


6.3. Toggle the enable state of a database comparison pair

PATCH

https://<servername:port>//abs/api/databaseComparePair/A

Parameter

```
dbPairName = A //Name of the database pair
```

This call returns the updated database pair.

This method toggles the enabled value of the database pair and is a simple way to change the state in the background.

Example


```
{

  "kind": "RESULT",
"request": "abs#createDatabaseComparePair",
"status": "OK",
"item_count": 1,
"items": [ {
      "kind": "DATABASE",
      "name": "A",
      "user": null,
      "description": null,
      "sourceName": "test_mv",
      "sourceVersion": 1,
      "targetName: "test_mv",
      "targetVersion": 2,
      "enabled": true
} ]
}
```


6.4. Delete a database comparison pair

DELETE

/api/databaseComparePair

https://<servername:port>/abs/api/databaseComparePair

Parameter

```
sourceDatabaseName  = test_mv   //Name of the source database

  sourceVersion        = 1        //Version of the source database

  targetDatabaseName  = test_mv   //Name of the target database

  targetVersion       = 2         //Version of the target database
```

https://<servername:port>/abs/api/databaseComparePair?
sourceDatabaseName=test_mv&sourceVersion=1&targetName=test_mv&targetVersion=2

Example


```
{

  "kind": "RESULT",
```

```
"request": "abs#deleteDatabaseComparePair",
"status": "OK",
"item_count": 0,
"items": []
}
```


This call returns **null** if everything is correct. Otherwise, all errors will be returned. The given parameter searches and deletes all pairs. If the databases are the same, source and target are exchangeable. All found database pairs are deleted even if it should just be one in the first place.

DELETE

/api/databaseComparePair/{dbPairName}

http://<servername:port>/abs/publishing/databaseComparePair/A

Parameter

```
databasePairName = A    //Name of the database pair
```

Deletes a database pair with the pair name. Only one database name is required for this call. This call returns **null** if everything is correct. Otherwise, all errors are returned. The given parameter searches and deletes all pairs.

Example


```
{

  "kind": "RESULT",
"request": "abs#deleteDatabaseComparePair",
"status": "OK",
"item_count": 0,
"items": []
}
```

7. Get an item

The ARIS API offers to retrieve groups, objects (definitions) and models. The desired item must be identified by its ARIS GUID or a full CONNECT item-id (for example, **c.process.Sample DB.CibrcP1SEdsnKQALzQzOTg.-1**).

Identifying via group path + name is unsupported as this is possibly ambiguous.

Example

GET

```
https://<servername:port>/abs/api/groups/Sample%20DB/
1191ae90-02f7-11dc-2729-000bcd0cce4e
```

By default, only the name attribute (AT_NAME) is included in the response. If you need more attributes, the URL parameter **attributes** must be given. Possible values are

- **all**: all non-empty attributes
- a comma-separated list of attribute type numbers, API names or type-GUIDs (all of these can be arbitrarily mixed)

  Example: attributes = 1, AT_DESC, AT_AUTH // attribute name, description, author

Remarks

- Connections are not supported.
- In case of versioned databases it is possible to retrieve a specific version via the parameter **changelist**. Possible values are -1 (workspace, default), 0 (last submitted version) or a specific change list number.

Note: It is also possible to retrieve HTTP portal links that can be directly used in the browser to access the item via the CONNECT portal. Add URL parameter **withportallinks = true**.

Portal Link Examples (for a model)

```
{
"kind": "LINK",
"method": "GET",
"href": "https://<servername:port>/#default/item/c.process.Sample
DB.CibrcP1SEdsnKQALzQzOTg.-1",
"rel": "ITEM_MODEL"   // CONNECT Item View
},
```

```
{
"kind": "LINK",
"method": "GET",
```

```
"href": "http://<servername:port>/#default/repository/a.model.Sample
DB.CibrcP1SEdsnKQALzQzOTg.-1",
"rel": "REPO_MODEL"   // CONNECT Repository View
},
```

```
{
"kind": "LINK",
"method": "GET",
"href": "https://<servername:port>/#default/thinclient/c.process.Sample
DB.CibrcP1SEdsnKQALzQzOTg.-1",
"rel": "TC_MODEL"   // CONNECT Designer View
},
```

```
{
"kind": "LINK",
"method": "GET",
"href": "http://
<servername:port>/abs/downloadClient/aris_database.jsp?configuration=ARIS&apps
erver=myserver&database=Sample%20DB&guid=0a26eb70-fd52-11db-2729-
000bcd0cce4e&language=en_US&tenant=default",
"rel": "DC_MODEL"   // CONNECT Download client
}
```

8. Get group children

A group usually contains children, that is, subgroups as well as models and objects.

GET

```
https://<servername:port>/abs/api/groups/Sample%20DB/4a713de0-5d02-11e3-0fda-fd81e986d7e2/children
```

This call returns all subgroups of the group identified by the given GUID or CONNECT item-ID. If you additionally need the models and/or objects in the group, you can add the URL parameters **withmodels=true** or **withobjects=true**.

GET

```
https://<servername:port>/abs/api/groups/Sample%20DB/4a713de0-5d02-11e3-0fda-fd81e986d7e2/children?withmodels=true&withobjects=true
```

Remark

It also possible to retrieve an entire subtree by passing the URL parameter **recursive=true**.

Note

There is an absolute result size limit of 10,000 items (regardless of the page size you choose and how many paging requests are sent).

If parameter withobjects=true and withmodels=false is used for retrieving subgroups and objects, the limit is 100,000. If a group contains more than

- 10,000 subgroups or
- more than 100,000 groups and objects (**withobjects=true** and **withmodels=false**),

you will not be able to retrieve them all.

The maximum number of subgroups that can be recursively retrieved is 10,000.

Depending on the ARIS license, other size limits may apply.

9. Get content of a model

The content of a model consists of occurrences (= model objects, model connections). They can be retrieved in a **getModel** call by passing the URL parameter **withcontent=true**.

GET

```
https://<servername:port>/abs/api/models/Sample%20DB/88ba40a0-
cfb6-11e0-2556-5c260a398437?withcontent=true
```

Remarks

Model objects/model connections are a blend of data from the occurrence and definition levels. They do not contain any graphical information, such as dimension (width, height), and no coordinates of positions and connection paths.

Example for model object item

```
{
"kind": "MODELOBJECT",
"occid": "(7wLHfY8btgy:u:L+6PmFOGfzQQ4:x:L+33+c)",
"guid": "d734eb6f-cf14-11e0-2556-5c260a398437",
"link": {
       "kind": "LINK",
       "method": "GET",
       "href": "https://<servername:port>/abs/api/objects/Sample
DB/d734eb6f-cf14-11e0-2556- 5c260a398437?language=en_US&methodfilter=dd838074-
ac29-11d4-85b8- 00005a4053ff",
       "rel": "OBJECT"
       },
"type": 239,
"typename": "Strategy",
"apiname":"OT_STAT",
"symbol": 1627,
"symbolname": "Tactic",
symbol_apiname": "ST_TACTIC",
"attributes": [
       {
       "kind": "ATTRIBUTE",
       "id": "6S4A4i43Hh0:p:L=1=1033:1:s",
       "typename": "Name",
       "type": 1,
       "apiname": "AT_NAME",
       "language": "en_US",
       "value": "Reduce operational costs in Supply Chain"
       }
     ]
}
```

Example for model connection item


```
{
"kind": "MODELCONNECTION",
"occid": "(7wLHfY8btgy:u:L+-6xMo0A_0oEz:y:L+34+c)",
"type": 67,
"typename": "encompasses",
"apiname": "CT_SUBS_1",
"source_guid": "c7ca78b0-abcf-11e0-7ee8-5c260a398437",
"target_guid": "289b0560-ac64-11e0-7ee8-5c260a398437",
"source_link": {
"kind": "LINK",
"method": "GET",
"href": "https://<servername:port>/abs/api/objects/Sample DB/c7ca78b0-
abcf-11e0-7ee8-5c260a398437?language=en_US&methodfilter=dd838074-ac29-11d4-
85b8-00005a4053ff",
        "rel": "OBJECT"
},
"target_link": {
       "kind": "LINK",
       "method": "GET",
       "href": "http://<servername:port>/abs/api/objects/Sample
DB/289b0560-ac64-11e0-7ee8- 5c260a398437?language=en_US&methodfilter=dd838074-
ac29-11d4-85b8- 00005a4053ff",
       "rel": "OBJECT"
       },
"source_occid": "(7wLHfY8btgy:u:L+-7cJgUsuG4Td:x:L+33+c)",
"target_occid": "(7wLHfY8btgy:u:L+-2apRUFiN0WT:x:L+33+c)"
}
```

10. Create an item

The API offers support for creating groups and objects. For objects, groups and models, it is possible to create attributes (or more correctly formulated in ARIS terminology: it is possible to maintain attributes with a value). You can add multiple attributes in the same call. For all added attributes, the same database language will be used (from URL parameter **language**).

Furthermore, it is possible to create an assignment relationship between an existing object and an existing model.

Note:

- The creation of models is not supported.
- For already existing objects, groups, and models, it is possible to add new attributes. [29]

Example

Create a new group with name "My Subgroup" in the main group (GUID = 4a713de0-5d02-11e3-0fda-fd81e986d7e2)

POST

```
https://<servername:port>/abs/api/groups/United%20Motor%20Group?
language=en&parent=4a713de0-5d02-11e3-0fda-fd81e986d7e2
```

Request body:

```
{
"kind": "GROUP",
"attributes": [
      {
      "kind": "ATTRIBUTE",
      "type": "AT_NAME",
      "value": "My Subgroup"
      }
      ]
}
```

Example

Create a new object of type

43

(OT_ORG_UNIT) with two attributes (AT_NAME and AT_DESC). Database language = English; parent group = 4a713de0-5d02-11e3-0fda-fd81e986d7e2 (= main group)

POST

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group?
language=en&parent=4a713de0-5d02-11e3-0fda-fd81e986d7e2
```

Request body:

```
{
"kind": "OBJECT",
"type": 43,
"attributes": [
      {
      "kind": "ATTRIBUTE",
      "type": "AT_NAME",
      "value": "My first OrgUnit "
      },
      {
      "kind": "ATTRIBUTE",
      "type": "AT_DESC",
      "value": "This is the long description"
      }
      ]
}
```

Remarks

The **kind** entries in the request body are redundant and not required.

There is a slight difference in the behavior whether creating an object or a group with a duplicate name.

- If you create a new group with the name **X** and there is already a group **X** in the desired parent group, then the new group will automatically be renamed to **X(1)**.
- If you create a new object with name **X** and there is already an object **X** in the desired parent group, then the new object has the name **X**, so there are two objects with the name **X**.

11. Create model content

Models cannot be created via the ARIS Repository API, but you can create model objects and model connections. When creating a new connection, it is possible to use objects that are created in the same request. As they do not have an ID yet, you must define a temporary occ-id (starting with #) and use these temporary occ-ids when defining source and target of the new connection.

Example

PUT

```
https://<servername:port>/abs/api/models/United%20Motor%20Group/
42ad5380-3d0e-11e5-6479-22000b630ca4
```

Request body:

```
{
"modelobjects": [
      {
      "kind": "MODELOBJECT",
      "occid": "#1",
      "type" : 22,
      "symbol": 335,
      "attributes": [
            {
            "kind": "ATTRIBUTE",
            "type": 1,
            "value": "new function via model update"
            }
            ]
      },
      {
      "kind": "MODELOBJECT",
      "occid": "#2",
      "type" : 18,
      "symbol": 1,
      "attributes": [
            {
            "kind": "ATTRIBUTE",
            "type": 1,
            "value": "new event via model update"
            }
            ]
      }
      ],
"modelconnections": [
      {
      "kind": "MODELCONNECTION",
      "type": 44,
      "source_occid": "#1",
      "target_occid": "#2"
```

```
        }
        ]
}
```

Remarks

- When creating objects as shown above, a new definition object is created. If you want to reuse an already existing object, provide the GUID, for example,

```
        {
        "kind": "MODELOBJECT",
        "occid": "#1",
        "guid" : "4686bd20-3d0e-11e5-6479-22000b630ca4"
        "type" : 22,
        "symbol": 335
        }
```

- As model objects and model connections are created without any positional information, they will be located one above the other in the upper left corner of the model. It is possible to enforce an automatic layout of the model when it is opened for the first time in ARIS Architect/Designer by passing the URL parameter **layoutonopen=true**.

Note: A user with **ARIS API Mobile Access** license can only create one model object or one model connection in a single request.

12. Delete model content

Model objects or model connections can be deleted from a model via their occ-ids. If you delete a model object, then all affected model connections are deleted automatically (no need to include them in the request).

DELETE

```
https://<servername:port>/abs/api/models/United%20Motor%20Group/
1672a301-3d14- 11e5-6479-22000b630ca4/objects?
occid=(-7O3yxAzzRgN%3Au%3AL%2B-7RaSdNnA6hr%3Ax%3AL%2B33%2Bc)
```

Remarks

- Occ-IDs often contain special characters. It is important to ensure properly urlencoded values.
- In the request path, make sure to have **/objects** or **/connections** at the end, otherwise the model itself will be deleted!

Note: A user with **ARIS API Mobile Access** license can only delete one model object or one model connection in a single request (automatically deleted model connections are not counted).

13. Move an item

It is possible to move an existing model, group, or object from its current location (i.e. parent group) to some other group. Exception: the main group of a database cannot be moved.

The move can be done by an update operation (PUT) and passing the URL parameter **PARENT**, along with the GUID of the desired new parent group.

Note: The creation of models is not supported.

Example

Move the model identified by its GUID 31c0dc4d-467f-11d4-bb1d-00105a0ef4f4 to the group with GUID 5ba90461-7ec4-11e3-01db-b51dcc951f78.

PUT

https://<servername:port>/abs/api/models/United%20Motor%20Group/ 31c0dc4d-467f-11d4-bb1d-00105a0ef4f4?parent= 5ba90461-7ec4-11e3-01db-b51dcc951f78

Request body:

{} // or any desired additional update-data e.g. for attributes

14. Attributes

Attributes are no top-level items and are tightly bound to their parent item (database, object, group, model). You can retrieve attributes only along with its parent item to which they belong.

Attributes can be identified by an integer type number (for example, 1), the API name (for example, AT_NAME) or a type GUID. These type identifiers can be freely mixed in the same request and even in the same parameter, for example, in the URL parameter **attributes**.

Attributes are maintained in the database language as given by the URL parameter **language**. If no such parameter is given, the default language of the current database is used.

Remarks

- Currently, only a limited subset of attribute types is supported, especially integer, float, text, timestamp, time, value attributes, binary.
- Binary data must be sent as BASE64 encoded string.

14.1. Retrieval

You cannot retrieve an attribute on its own. You must retrieve the parent item (for example, an object or a model to which the attribute belongs) by using an appropriate get-request and specifying the attribute type(s) (URL parameter attributes) you want to have in the response. If no attribute type is given, the value of the **Name** attribute (AT_NAME) is returned by default (but not in case of a database).

Examples

Get the model with GUID **bf2a9d60-7cb8-11dc-2729-000bcd0cce4e** with all maintained attributes.

GET

```
https://<servername:port>/abs/api/models/Sample%20DB/
bf2a9d60-7cb8-11dc-2729-000bcd0cce4e?language=en&attributes=all
```

Get the model with the attributes **Name** (AT_NAME, type number 1), **Description** (AT_DESC), and **Status** (AT_STATE_1). Attributes can be specified by type numbers or API names/type GUIDs. These can be freely mixed in a comma-separated list.

GET

```
https://<servername:port>/abs/api/models/Sample%20DB/
bf2a9d60-7cb8-11dc-2729-000bcd0cce4e?language=en&attributes=1,AT_DESC,AT_STATE_1
```

Get the details of the database **Sample DB** including all maintained attributes.

GET

```
https://<servername:port>/abs/api/databases/Sample%20DB?
language=en&attributes=all
```

14.2. Creation

You can create an attribute for all items (except databases) together with its parent in the same request.

Example

Create an object with the two attributes **name** and **description**.

POST

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group?language=en
&parent=4a713de0-5d02-11e3-0fda-fd81e986d7e2
```

Request body:

```
{
"kind": "OBJECT",
"type": 43,
"attributes":
      [
      {
      "kind": "ATTRIBUTE",
      "type": "AT_NAME",
      "value": "I am an OrgUnit!"
      },
      {
      "kind": "ATTRIBUTE",
      "type": 9,
      "value": "This is the long description"
      }
      ]
}
```

14.3. Updating or creation for existing item

If you want to add a new attribute to an already existing item or if you want to update the value of an existing attribute, you must do an update operation (PUT) on the parent item (not supported for a database item).

Example

Update the name of an existing object and add the author attribute.

PUT

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group/
b7e90c56-00cf-11e2-21d1-5c260a628455?language=en
```

Request body:

```
{
"attributes": [
        {
        "kind": "ATTRIBUTE",
        "type": 1,
        "value": "Just renamed it"
        },
        {
        "kind": "ATTRIBUTE",
        "type": "AT_AUTH",
        "value": "I am the author"
        }
        ]
}
```

14.4. Deleting an attribute

Deleting an attribute cannot be done via an update operation on the parent item. Instead, the dedicated DELETE operation must be used (not supported for a database item). The type numbers of the attributes to be deleted must be passed via the URL parameter **typenumbers**.

DELETE

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group/b7e90c56-00cf-
11e2-21d1-5c260a628455/attributes?language=en&typenumbers=AT_DESC, AT_AUTH
```

Remarks

- Do not forget to have **/attributes** at the end of the URL path, otherwise you do a DELETE on the parent object and it is gone!
- The result of a successful delete operation is simply empty with STATUS = OK, for example,

```
{
"kind": "RESULT",
"request": "abs#deleteObjectAttributes",
"status": "OK",
"item_count": 0,
"items": []
}
```

Note: Only one attribute can be deleted in a single request for users with an **ARIS API Mobile Access** license.

14.5. Styled values for text attributes

In ARIS, text attributes (for example, AT_DESC) can also have a styled value. If you want to retrieve text attributes with their styled value, you must set the URL parameter **withstyledtext = true**. If the text attribute has got a valid styled value, it will be returned in addition to the standard value field.

A typical response would be as follows:

```
       {
       "kind": "ATTRIBUTE",
       "id": "2Nc1glOg_LN:p:L=9=1033:1:s",
       "typename": "Description/Definition",
        "type": 9,
       "apiname": "AT_DESC",
       "language": "en_US",

       "styled_value":
              "<html><body>
              <p       style=\"margin-left:0;margin-bottom:0;
                            margin-top:0\"><b>my&#32;updated</
b> description
              </p>
       </body></html>",

       "value": "my updated description"
       }
```

The styled value is HTML starting with tags **<html><body>**.

It is also possible to set a styled value when creating or updating an attribute. Simply pass "**styled_value**" with a valid HTML string.

Remarks

- The URL parameter **withstyledtext** is not required for write operations (POST, PUT).
- If a write operation (POST, PUT) contains both **value** and **styled_value**, then **styled_value** has precedence and **value** is basically ignored. According to standard ARIS behavior, **value** will then contain a plain-text representation of the HTML from **styled_value**.
- Only a basic set of HTML is supported: <b>, <i>, <u>, <strike>, <ul>, <ol>, <li>, <br>, <p>, , .
- For styling information (margin, color, font-family, size), use an inline style attribute, for example,

```
<span style=\"font-size:48pt;\">

          <span style=\"font-family:Kokila;\">Hello World</span>

      </span>
```

- Unsupported HTML tags are silently ignored (only their text content will be considered).
- Make sure to correctly escape JSON special characters, for example, **\n** must be escaped as **\\n** and " as **\"**.

- If the HTML in **styled_value** does not contain any real formatting, a plain text value is set, for example,

```
<html><body><p> My name is Bond. James Bond.</p></body></html>
```

Result

"value" : "My name is Bond. James Bond."

"styled_value" : null

15. Assignments

An assignment is simply a link between an existing object and an existing model. With the ARIS Repository API, you can create and delete assignments. It is not possible to retrieve a specific assignment; assignments are always included automatically in the response when you retrieve an object.

Example

Create an assignment

POST

```
https://<servername:port>/abs/api/objects/Sample DB/6aa1b167-
fac0-11de-55c7-001a6b3c820f/assignment/a39aefb0-fa1c-11db-2729-000bcd0cce4e
```

objectGUID (source of assignment): 6aa1b167-fac0-11de-55c7-001a6b3c820f

modelGUID (target of assignment): a39aefb0-fa1c-11db-2729-000bcd0cce4e

Example

Delete an assignment

DELETE

```
https://<servername:port>/abs/api/objects/Sample DB/6aa1b167-
fac0-11de-55c7-001a6b3c820f/assignment/a39aefb0-fa1c-11db-2729-000bcd0cce4e
```

16. Variants

An existing variant relationship between models or objects is part of the standard response when retrieving a model or object (creation or deletion of variant relationships is currently not supported).

The variant master data contains a property **variant_guid** with an array of GUIDs which represent the linked variants.The GUID of the variant master will in turn appear in the variant data as property **variant_masterguid**.

Example 1.

The model **Sales order processing (as-is)** has a variant named **Sales order processing (to-be)**. When retrieving these models, the response data will look as follows:


```
{
      "kind": "MODEL",
      "guid": "0adfca71-2960-11dc-2729-000bcd0cce4e",
      "type": 13,
      "typename": "EPC",
      "apiname": "MT_EEPC",
      "attributes": [
        {
          "kind": "ATTRIBUTE",
          "typename": "Name",
          "type": 1,
          "apiname": "AT_NAME",
          "language": "en_US",
          "value": "Sales order processing (as-is)"
        }
      ],
      "variant_guids": [
      "45ad5550-f97c-11db-2729-000bcd0cce4e"
      ]
 }

 {
      "kind": "MODEL",
      "guid": "45ad5550-f97c-11db-2729-000bcd0cce4e",
      "type": 13,
      "typename": "EPC",
      "apiname": "MT_EEPC",
      "attributes": [
        {
          "kind": "ATTRIBUTE",
          "typename": "Name",
          "type": 1,
          "apiname": "AT_NAME",
          "language": "en_US",
          "value": "Sales order processing (to-be)"
        }
      ],
      "variant_masterguid": "0adfca71-2960-11dc-2729-000bcd0cce4e"
}
```

17. Model graphic

The ARIS API does not provide positional information when retrieving a model with its contents. But it is possible to get a model graphic. It has format PNG (Portable Network Graphics) and is returned as a BASE64-encoded string. There are two optional URL parameters **maxwidth** and **maxheight** that allow you to control the size (in image pixels) of the generated graphic.

Example

GET

```
https://<servername:port>/abs/api/models/United%20Motor%20Group/0a26eb70-fd52-
11db-2729-000bcd0cce4e/graphic
```

18. Finding items in the database

It is possible to search for items according to several criteria that are AND-linked, that is, they reduce the number of matching result items. You must pass the URL parameter kind to declare what to search (model or object). In addition, you can define the desired type (= comma-separated list of object type numbers or model type numbers; API names or type GUIDs are also supported) and optionally a filter criterion with respect to an attribute.

Example

Find all models of type **13** (MT_EEPC), whose names start with **Sale** (written with small or large **s**).

GET

```
https://<servername:port>/abs/api/databases/United%20Motor%20Group/find?
kind=MODEL&typefilter=13&language=en&attrfilter=AT_NAME=Sale
```

Remarks

- Available item kinds: MODEL, OBJECT, GROUP
- For all attributes: + (isMaintained), - (isNotMaintained)
- For numbers, the usual operators are available: =, !=, >, <, >=, <=
- For Boolean: = and !=
- For text attributes: =
- The parameter **attrfilter** may contain fully parenthesized expressions (parenthesis= { }), combined with operators AND, OR.
- Date values must be given in UTC in the RFC 3339 Internet format YYYY-MM-DD, for example, 2021-10-15
- Timestamp values must be given in UTC in the RFC 3339 Internet Zulu time format YYYY-MM-DD'T'HH:MM:SS'Z', for example, 2021-10-15T08:22:55Z

Examples for timestamp queries

All objects changed in a given period of time

GET

```
https://<servername:port>/abs/api/databases/United%20Motor%20Group/find?
kind=OBJECT&language=en_US&attributes=all&attrfilter={AT_LAST_CHNG_2 >=
2021-10-15T00:00:00Z} AND {AT_LAST_CHNG_2 <= 2021-10-15T09:00:00Z}
```

All objects created in a given period of time

GET

```
https://<servername:port>/abs/api/databases/United%20Motor%20Group/find?
kind=OBJECT&language=en_US&attributes=all&attrfilter={AT_CREAT_TIME_STMP >=
2021-10-15T00:00:00Z} AND {AT_CREAT_TIME_STMP <= 2021-10-15T09:00:00Z}
```

Searching for a multi-value attribute

In the parameter **attrFilter**, you can use the operator **=** to find all items where at least one of the allowed values of the attribute is given.

Use the operator **!=** for all items where the given value is not maintained.

Example 2.

Assume a multi-value attribute 'Colours' (ApiName 348cf080-6099-11f0-6cfb-ac74b1271078) with possible values 'Red', 'Green', and 'Blue'.

The search expressions could look as follows:

```
attrfilter=348cf080-6099-11f0-6cfb-ac74b1271078 = Red //all items where 'Red'
is present
attrfilter=348cf080-6099-11f0-6cfb-ac74b1271078 = Red,Blue // 'Red' OR 'Blue'
is present
attrfilter=348cf080-6099-11f0-6cfb-ac74b1271078 != Green // all items without
'Green'
```

Remarks

- For the operator **!=** only a single value may be given, no comma-separated list of values.
- As for all other search queries you can alternatively use the ApiName (GUID) of the desired attribute value (instead of the localized name as shown above).

Searching for text: examples for the value of attrfilter

```
AT_NAME =       Jones
9              = Human Resources       // (9 = AT_DESC)
AT_AUTH -                              // (AT_AUTH not maintained)
{AT_NAME = Jones} AND {AT_DESC = Human Resources}
{{AT_NAME = Jones} OR {AT_DESC = Human Resources}} AND {AT_AUTH -}
```

- **Important**: in case of a parenthesized expression, the search value must not contain **{ or }**. If the search value must contain a curly parenthesis, pass an escaped character, for example, **\}** when searching for text attributes.
- The normal search behavior is similar to that of standard search engines: Contents of text attributes are token-based, that is, typical separators, such as blanks or dashes (-), are ignored. If the text attribute contains tokens from the search value (regardless of order/position), this is considered a match.

  Example: Search expression AT_NAME = holder certificate

  will find **Holder Certificate** and **certificate holder** and **the certificate holder**.

- URL parameter **matchcase**: if true, searching will be case-sensitive and order of tokens is relevant
- URL parameter **exactsearch**: if true, only exact matches are returned; special characters and notably the asterisk (*) are interpreted as simple character; in addition, the order of the tokens is important.

  Example: Search expression AT_NAME = holder certificate

  will find **Holder Certificate** but not **certificate holder** and not **The Holder Certificate**.
- Special search patterns for text (parameter **exactsearch** must be false):
  - **A*** returns all strings that start with **A** (**A** must be a string of length >=2, the first two characters must be letters or digits).
  - **A*B** returns all strings that start with **A** (**A** must be a string of length >=2, the first two characters must be letters or digits) and end with **B**.

Important note

The search functionality described above is not intended to load large amounts of data from an ARIS repository. Therefore, there is a result size limit (typically 100,000 items for kind=OBJECT, otherwise 10,000; both limits are also depending on the user license). This limit is absolute, that is, it is independent on the page size you choose and how many paging requests are sent.

Example: If a search query is sent to get all models of **EPC** type and the database contains 15,000 such models, then only the first 10,000 hits can be retrieved (for example, by sending 50 paging requests with page size 100).

In general, a search request that leads to several thousand hits should always be refined with more concrete search criteria.

19. Paging

Typically, when using the database-find operation from the previous chapter, many items will be found and returned to the caller. A default page size of 100 is in effect. A different page size can be indicated with the URL parameter **pagesize**. The maximum allowed page size depends on the user's license and server settings.

If there are more items to be returned than the page size allows, the response will contain an entry **next_pagetoken**. This indicates to the client that there is more data to load.

Example

Request

```
https://<servername:port>/abs/api/databases/United%20Motor%20Group/find?
kind=MODEL&language=en&attrfilter=AT_NAME&wildcards=true&attrcriterion==S*
```

Response

```
{
"kind": "RESULT",
"request": "abs#find",
"status": "OK",
"item_count": "100",
"next_pagetoken": "1:100:guid",
"items": [
…       100 items here
]
}
```

In order to get the next page of results, the client must send **the same request again and add the page token**, for example, as query parameter.

GET

```
https://<servername:port>/abs/api/databases/United%20Motor%20Group/find?
kind=MODEL&language=en&attrfilter=AT_NAME&wildcards=true&attrcriterion==S*&paget
oken=1:100:guid
```

Response

```
{
"kind": "RESULT",
```

```
"request": "abs#find",
"status": "OK",
"item_count": "35",
"items": [
… remaining 35 items here
]
}
```

In the above example, the second response contains no entry **next_pagetoken** --> the client knows that there are no more items to load.

Notes

- The maximum allowed page size is 10,000 for object items and 1,000 for all other items like models, groups, etc. If a page size is given that is too large, it will be silently corrected to the allowed maximum.
- Regardless of paging, there is an **absolute size limit** for the results depending on the user license, typically 100,000 for object items and 10,000 for all other items like models, groups, etc. For example, if you use page size 10,000 for getting object items or 1,000 for getting models, at most 10 paging requests will return data. If the total result set is larger, it is not possible to retrieve the remainder.

20. Ordering the result

If more than one item is returned by a request (for example, database-find), the items are ordered by their GUID in ascending order. Ordering by GUID itself is normally useful but this ensures a stable result, especially with respect to paging.

You can pass the URL parameter **orderby** in order to indicate a different sorting, for example,

```
…&orderby = name      (order by attribute AT_NAME, ascending)
…&orderby = -name     (order by attribute AT_NAME, descending)
```

Other possible values:

- **modified** (ascending)
- **-modified** (descending)
- **path** (group path starting at main group)

21. Generic queries

Generic queries attempt to provide a kind of graph-based retrieval of items. Syntax and functionality are similar to the XML-based transformation configuration used in ARIS. Familiarity with these transformation configurations might therefore be helpful in understanding the use of the generic queries.

The general idea is as follows:

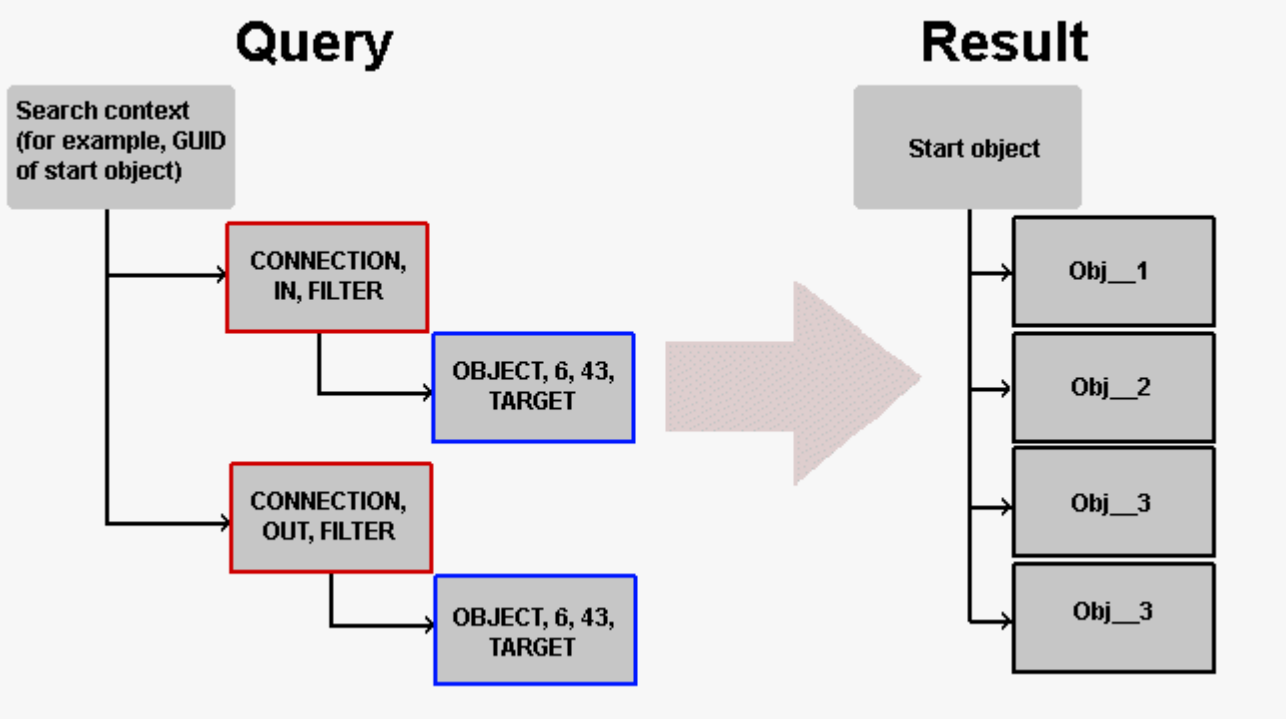
- A search context must be specified with which the query should start. The search context can be a list of specific objects or models (defined as a comma-separated list of GUIDs, for example, "start_guids" = "id1, id2, id3") or a list of object or model type numbers ("start_types" = "6, 43, 78").
- If the search context is made up of objects, the endpoint must be: /abs/api/objects/<db name>/query
- If the search context is made up of models, the endpoint must be: /abs/api/models/<db name>/query
- The search context must consist of GUIDs or type numbers representing either objects or models. Mixing objects and models is not allowed.
- Filter items that should be returned as result must be marked as TARGET.
- Non-TARGET filter items can be marked as FILTER for clarity (not required).
- Starting from the items defined by the search context, it is possible to find and retrieve objects or models that can be reached from the start items through a sequence of filter items. Each sequence is a nested path of items. Only one item of the TARGET function is allowed within a single path.
- Available properties for filter items:

  type MODEL, OBJECT, CONNECTION, ASSIGNMENT, OCCURRENCE, CONNECTIONOCCURRENCE

  function FILTER, TARGET

  direction IN, OUT (only allowed for items with type CONNECTION)

  typenum comma-separated list of types
- Type information (for the item property "typenum") can be numbers, API names, or type-GUIDs. It is usually most efficient to use type numbers (integers). Use API names (for example, OT_FUNC) if you want to make the JSON more human readable, or a type-GUID in case of user-defined types.

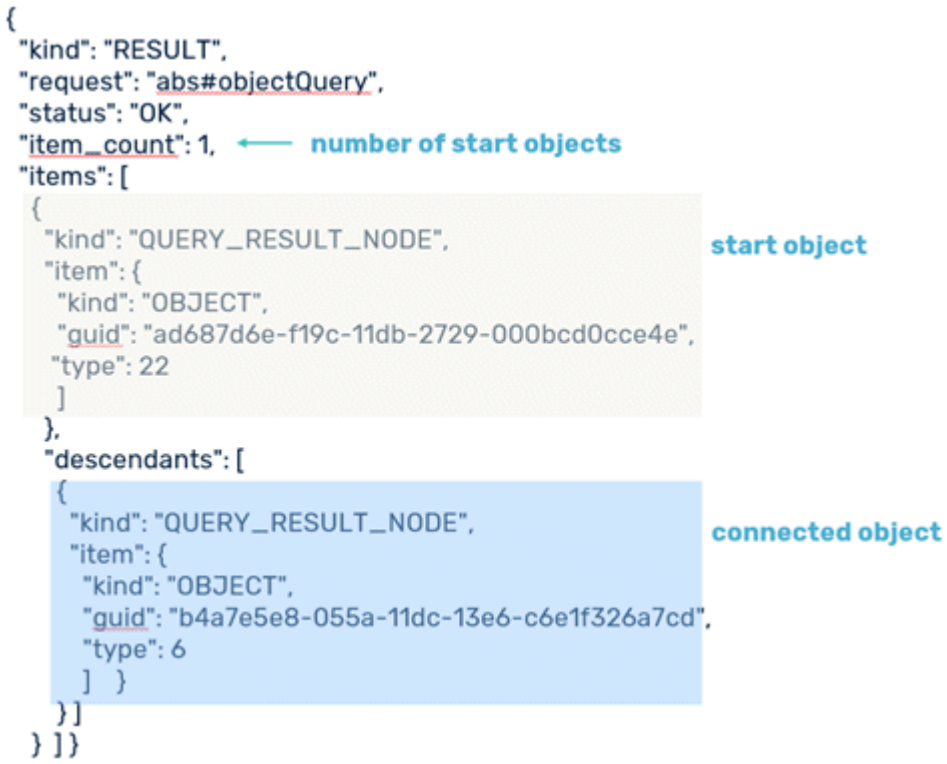


Query with two parallel filter paths: the upper path returns all objects of type 6 or 43 to which the start object has an incoming connection; the lower path returns all objects of type 6 or 43 to which the start object has an outgoing connection.

Sample request body:

```
{
  "start_guids" : "ad687d6e-f19c-11db-2729-000bcd0cce4e",
  "items" : [
    { "type" : "CONNECTION",
      "direction" : "IN",
        "items" : [
        { "type" : "OBJECT",
          "typenum" : "6,43",
          "function" : "TARGET"
        }
      ]
    },
    { "type" : "CONNECTION",
      "direction" : "OUT",
        "items" : [
        { "type" : "OBJECT",
          "typenum" : "6,43",
          "function" : "TARGET"
        }
      ]
    }
  ]
}
```

Sample response (slightly shortened)



Remarks

- Not all possible combinations of filter items lead to useful results. Only the use-case examples listed below are officially supported. Other combinations might not work.
- Although a generic query is of read-only nature, it must be sent as a POST request because the query definition must be provided in the request body.
- The request body must contain a JSON structure with the query, which is a nested list of filter items.
- Start items appear in the result as top-level entries (that is, the **item_count** in the response reflects the number of these top-level items, not the number of all objects found that match the filter criteria).
- The descendants of each top-level node represent the TARGET objects that meet the filter criteria; this list is empty if there was no match for the respective top-level item (= start item).

21.1. Use case: Get connected objects (def level)

If you want to determine all objects of a type that have a connection to a given object (that is, the start object) on definition level, you can use a CONNECTION filter item. Please note that you must specify the respective connection direction IN or OUT (as seen from the start object). If you do not care about the direction, you must define two filter paths, one for IN, the other for OUT, as shown below. The objects of interest to be returned must be specified by at least one type information and marked as TARGET.

Example

POST

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group/query
```

Request body:

```
{
  "start_guids" : "ad687d6e-f19c-11db-2729-000bcd0cce4e",

  "items" : [
    { "type" : "CONNECTION",
      "direction" : "IN",
      "items" : [
            { "type" : "OBJECT",
            "typenum" : "6,43",
            "function" : "TARGET"
        }
    ]
    },

    { "type" : "CONNECTION",
      "direction" : "OUT",
     "items" : [
        { "type" : "OBJECT",
          "typenum" : "6,43",
          "function" : "TARGET"
        }
      ]
    }
  ]
}
```

21.2. Use case: Get connected objects (occ level)

Queries usually operate on definition level. If you want to retrieve data based on filtering in terms of occurrences, you can proceed as in the following example. Use filter items of type OCCURRENCE and CONNECTIONOCCURRENCE to indicate the jump from the definition level to the occurrence level. Within the CONNECTIONOCCURRENCE item, define the desired connection and target object.

Example

POST

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group/query
```

Request body:

```
{
   "start_guids":"ad687d6e-f19c-11db-2729-000bcd0cce4e",
   "items":[
      {
         "type":"OCCURRENCE",
         "items":[
            {
               "type":"CONNECTIONOCCURRENCE",
               "items":[
                  {
                     "type":"CONNECTION",
                     "direction":"IN"
                  },
                  {
                     "type":"OCCURRENCE",
                     "items":[
                        {
                           "type":"OBJECT",
                           "typenum":"6,43",
                           "function":"TARGET"
                        }
                     ]
                  }
               ]
            }
         ]
      }
   ]
}
```

The above query explicitly requires that there is a connection between the start object and the target object. Sometimes it may be sufficient for your needs that there is a connection on definition level, but you want the target object to have an occurrence, that is, that the target is used in a model. This can be formulated as shown in the following example. Note that the OCCURRENCE item must have a **typenum** property with the type of the symbol used for the object occurrence.

Example

POST

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group/query
```

Request body:

```
{
  "start_guids": "ad687d6e-f19c-11db-2729-000bcd0cce4e",
  "items": [
    {
      "type": "CONNECTION",
      "direction": "IN",
      "items": [
        {
```

```
          "type": "OBJECT",
          "typenum": "6",
          "function": "TARGET",
          "items": [
            {
              "type": "OCCURRENCE",
              "typenum" : "ST_APPL_SYS_TYPE",
              "function": "FILTER"
            }
          ]
        }
      ]
    }
  ]
}
```

21.3. Use case: Get models with occurrences of an object

A frequently needed question about an object is: In which models does it occur? This can be specified by using an OCCURRENCE filter item and a MODEL item as target. Please note that it is not possible to ask for all kinds of model: you must specify at least one model type as **typenum**.

Example

POST

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group/query
```

Request body:

```
  {   "start_guids":"6e285d97-b5a4-11e3-61a7-c1b8f41beb93",

     "items":[
      {
"start_guids" : "ad687d6e-f19c-11db-2729-000bcd0cce4e",
  "items" : [
    {
    "type" : "OCCURRENCE",
    "items" : [
          {
          "type" : "MODEL",
          "typenum" : "13, 134",
          "function" : "TARGET"
          }
       ]
    }
```

```
   ]
}
```

21.4. Use case: Get objects to which a model is assigned

If you want to find out for a certain model for which objects it is used as an assignment, then you can use the ASSIGNMENT filter item and an OBJECT as target item. The start item is the GUID of the model you are interested in.

Example

POST

```
https://<servername:port>/abs/api/models/United%20Motor%20Group/query
```

Request body:

{ "start_guids":"6e285d97-b5a4-11e3-61a7-c1b8f41beb93",

"items":[

```
      {
         "type":"ASSIGNMENT",
         "function":"FILTER",
         "items":[
            {
               "type":"OBJECT",
               "typenum":"22",
               "function":"TARGET"
            }
         ]
      }
   ] }
```

21.5. Use Case: Get models with occurrence pattern

It is possible to check for one or more given models whether a certain occurrence pattern is present, for example, whether there is a function object (type 22) with an incoming connection occurrence from an object of type 6 (application system type). The target objects are returned as descendants of the start models, that is, all top-level items in the response with an empty descendant list did not meet the pattern condition.

Example

POST

```
https://<servername:port>/abs/api/models/United%20Motor%20Group/query
```

Request body:

```
  {
"start_guids" : "45ad5550-f97c-11db-2729-000bcd0cce4e",
"items" : [
     {
       "type" : "OCCURRENCE",
       "items" : [
              {
              "type" : "OBJECT",
              "typenum" : "22",
              "function" : "FILTER"
              },
              {
              "type" : "CONNECTIONOCCURRENCE",
       "items" : [
                     {
                     "type" : "CONNECTION",
                     "function" : "FILTER",
                     "direction" : "IN",
                     "items" : [
                            {
                            "type" : "OBJECT",
                            "typenum" : "6",
                            "function" : "TARGET"
                            }
                            ]
                     }
              ]}
       ]}
   ]}
```

21.6. Use Case: Get models with occurrence pattern

If you want to get the assigned models of a specific object, it is recommended to use the getObject functionality (see Chapter Get an item [18]).

However, if you have a list of object GUIDs, it might be helpful to determine the assigned models with a single query that uses an ASSIGNMENT filter item and a MODEL item as the target (specify all desired model types in the property **typenum**).

Example

POST

```
https://<servername:port>/abs/api/objects/United%20Motor%20Group/query
```

Request body:

```
  {
 "start_guids" : "1055825e-f979-11db-2729-000bcd0cce4e, 10558255-
f979-11db-2729-000bcd0cce4e",

   "items": [
           {
            "type":"ASSIGNMENT",
            "items":[
                    {
                    "type":"MODEL",
                    "typenum":"14",
                    "function":"TARGET"
                    }
                ]
           }
     ]
}
```

22. Support and legal information

This section provides you with some general information regarding product support and legal aspects.

22.1. Documentation scope

The information provided describes the settings and features as they were at the time of publishing. Since documentation and software are subject to different production cycles, the description of settings and features may differ from actual settings and features. Information about discrepancies is provided in the Release Notes that accompany the product. Please read the Release Notes and take the information into account when installing, setting up, and using the product.

If you want to install technical and/or business system functions without using the consulting services provided by SAG ARIS GmbH, you require extensive knowledge of the system to be installed, its intended purpose, the target systems, and their various dependencies. Due to the number of platforms and interdependent hardware and software configurations, we can describe only specific installations. It is not possible to document all settings and dependencies.

When you combine various technologies, please observe the manufacturers' instructions, particularly announcements concerning releases on their Internet pages. We cannot guarantee proper functioning and installation of approved third-party systems and do not support them. Always follow the instructions provided in the installation manuals of the relevant manufacturers. If you experience difficulties, please contact the relevant manufacturer.

If you need help installing third-party systems, contact your local ARIS sales organization. Please note that this type of manufacturer-specific or customer-specific customization is not covered by the standard ARIS software maintenance agreement and can be performed only on special request and agreement.

22.2. Support

Product support

We provide support for ARIS products to all customers with a valid support contract.

Information on product version availability and support, as well as sustained support dates, is available at ARIS Community.

Contact our Global Support services at ARIS Community to raise and update support incidents, or post ideas and feature requests.

Community

Register with ARIS Community to download products, updates and fixes, find expert information, and interact with other users.

Documentation

For information on how to use and activate products and for other technical support questions, visit the ARIS Documentation website.

If you want to give feedback on any documentation topic, write to documentation@aris.com.

22.3. Legal notices

This document applies to ARIS Version 10 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2010-2026 SAG Aris GmbH, Saarbrücken, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name ARIS and all SAG Aris GmbH product names are either trademarks or registered trademarks of SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

The Software may contain or be distributed with certain commercial third party components (“Third Party Components”), which may be subject to special license terms and conditions (“Third Party Terms”) located at https://aris.com/terms.

Third Party Terms include important licensing and warranty information and disclaimers of third party licensors. For the avoidance of doubt, if Customer uses the Software in accordance with the Documentation, the Third Party Terms do not restrict the rights granted under this Agreement. In the event that a third party manufacturer no longer provides active support for any Third Party Component, Supplier will use commercially reasonable efforts to respond to any issues with the Software related to the Third Party Components.

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with SAG Aris GmbH and/or its subsidiaries and/or its affiliates and/or their licensors.