HP Asset Manager

Software Version: 9.50
Windows® operating system

Asset Manager 9.50 RESTful API

Document Release Date: March 2015
Software Release Date: March 2015

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Introduction 4

Resource types 6

RESTful API functions 7

Record 7

Create a record 7

Query a record 7

Query a record for a binary field 8

Update a record 8

Create a link between records 9

Delete a link from destination record 9

Delete a record 9

List 10

Query list 10

Query 1-N resources 11

Query URL parameters 12

Data types 14

Appendix. How to implement life cycle for work order with RESTful API 16

Send Documentation Feedback 19

Introduction

Since Asset Manager 9.50, a new set of industry standard-compliant RESTful APIs is introduced to help consume Asset Manager data upon which further business flows can be built.

These RESTful APIs offer the capability to manipulate Asset Manager records with create, update, delete, and other actions. These RESTful APIs are also friendly to any stateless application with JSON formatted data as transmission media.

Note: Although AM RESTful APIs are implemented in AM Web Service, it does not depend on any tagged Web Service, which means all the workflows and scripts triggered from these RESTful API calls are always the up-to-date version.

Note: In Asset Manager 9.30 and later versions, there also exists a set of old RESTful APIs (RESTful AQL). The new RESTful APIs introduced in Asset Manager 9.50 do not replace the old ones. Depending on your habits or experience, you can choose to use either the old RESTful AQLs or the new RESTful APIs. However, we recommend that you do not mix them in one application or program for maintenance consideration.

This document introduces the 9.50 version of the RESTful APIs. For more information about the old (9.30) version of the RESTful APIs, see the `Asset Manager 930 RESTful API (RestAQL)` white paper in the Asset Manager installation directory.

From the RESTful perspective, resources in Asset Manager can be categorized as data and functions.

- **Data:** Similar to the traditional RDBMS, Asset Manager has its own table schema and record data for each table. The RESTful APIs provide the basic database CRUDs. In addition, most resources in the database can be mapped directly to its schema. The RESTful APIs do not support schema functionalities.
- **Functions:** Functions stand for the additional user interactive components to manipulate the Asset Manager data. Functions include screen, view, wizard, statistics, and all other objects that can be found in the functional domain. Functional domain is one of the key data separation means in Asset Manager. Resources at all levels of functional domain are filtered out for users with various functional rights.

The Asset Manager 9.50 RESTful APIs have the following characteristics.

- The RESTful APIs implement CRUD functionalities.
- The RESTful APIs return JSON and binary data.
- The RESTful APIs leverage the existing BASIC authentication in the web service. In addition, the "black list" feature is enabled to filter sensitive data.

- The error handing and response code of the RESTful APIs follows the RESTful best practice.
- The RESTful APIs are empowered by AQL.

The following items are not supported by Asset Manager 9.50 RESTful APIs.

- Version information
- Transaction
- Cache
- Meta-data

Resource types

The following table shows the resource types that are supported by the Asset Manager 9.50 RESTful APIs.

Type	Descr.	Examples

Record	One record in a table.	<pre>{
		"Name": "Administrative Services",
		"CostCenter.Title": "HP Administrative",
		"CostCenter": {
		"ref-link": "db/amCostCenter/130681",
		"self": "HP Administrative"
		},
		"self": "Administrative Services, ",
		"Absences": {
		"ref-link": "db/amEmplDept/130165/Absences"
		},
		"ref-link": "db/amEmplDept/130165"
		}</pre>

List	Collection of records.	<pre>{
		"count": 127, //Total number
		"entities": [
		{
		"Name": "Administration", //Normal field
		"CostCenter.Title": "Corporate services", //1:1 field
		"CostCenter": { //1:1 link
		"ref-link": "db/amCostCenter/120711",
		"self": "Corporate services"
		},
		"bCreatedOnTheFly": { //System Itemized List
		"0": "No"
		},
		"MrMrs": "Mrs.", //Customized Itemize List
		"self": "Administration, ", //Display name
		"Absences": { //1:N link
		"ref-link": "db/amEmplDept/121238/Absences"
		},
		"ref-link": "db/amEmplDept/121238" //Detail link
		}
		]
		}</pre>

RESTful API functions

The Asset Manager 9.50 RESTful functions are summarized with samples in the this chapter.

Note: The samples provided in this section should be run in POSTMAN in Google Chrome. Before you run the samples, make sure that Asset Manager web service is started.

For Java samples, see "Appendix. How to implement life cycle for work order with RESTful API" on page 16.

Record

This section describes how to use RESTful API functions to perform record operations.

Create a record

POST /rs/db/<TableName>/

Response code: 201/400 for client error

Response body: A primary key in text

A sample is provided as follows.

URL: http://localhost:8080/AssetManagerWebService/rs/db/amEmplDept

Content-Type: application/json

Entity body:

```
{
  "name":"test",
  "title":"dev",
  "bCreatedOnTheFly": "1"
}
```

Query a record

GET rs/db/<TableName>/<PK>

Response code: 200 / 404 if not found

Response body: A JSON-formatted object.

A sample is provided as follows.

URL:
http://localhost:8080/AssetManagerWebService/rs/db/amEmplDept/130165?fields=CostCenter,CostCenter.Title,Absences,name

Content-Type: application/json

Response:


```
{
    "Name": "Administrative Services",
    "CostCenter.Title": "HP Administrative",
    "CostCenter": {
        "ref-link": "db/amCostCenter/130681",
        "self": "HP Administrative"
    },
    "self": "Administrative Services, ",
    "Absences": {
        "ref-link": "db/amEmplDept/130165/Absences"
    },
    "ref-link": "db/amEmplDept/130165"
}
```


Query a record for a binary field

GET rs/db/<TableName>/<PK>?fields=blbData

Response code: 200 / 404 if not found

Response body: Binary.

A sample is provided as follows.

URL: http://localhost:8080/AssetManagerWebService/rs/db/amImage/12345?fields=blbData

Note: In version 9.50, "application/octet-stream" is the only media type.

Note: You can only query a binary field. You cannot query both a common field and a binary field in one RESTful API.

Update a record

PUT rs/db/<TableName>/<PK>

Response code: 200 / 404 if not found

Response body: Primary key.

A sample is provided as follows.

URL: http://localhost:8080/AssetManagerWebService/rs/db/amEmplDept/12345

Content-Type: application/json

Entity body:

```
{
  "name":"new_name",
  "dLeave":1423463436234   //the number of milliseconds since January 1, 1970, 00:0
0:00 GMT
  "title": "new_title",    //Customized itemized list
  "bCreatedOnTheFly", {"1", "Yes"}  //System itemized list
}
```

Create a link between records

PUT rs/db/<Table Name>/<PK>/<Link Name>/<Reverse PK>

Response code: 200 / 404 if not found

Response body: Empty.

A sample is provided as follows.

URL: http://localhost:8080/AssetManagerWebService/rs/db/rs/db/amLocation/123/EmplDepts/456

Note: For an AM link, foreign key is always in the source table, and the other table is called the destination table. You can create a link from either table.

Delete a link from destination record

DELETE rs/db/<Table Name>/<PK>/<Link Name>/<Reverse PK>

Response code: 200 / 404 if not found

Response body: Empty.

A sample is provided as follows.

URL: http://localhost:8080/AssetManagerWebService/rs/db/rs/db/amLocation/123/EmplDepts/456

Note: For an AM link, foreign key is always in the source table, and the other table is called the destination table. You can delete a link from either table.

Delete a record

DELETE rs/db/<Table Name>/<PK>

Response code: 200 / 404 if not found

Response body: Primary key

A sample is provided as follows.

DELETE: http://localhost:8080/AssetManagerWebService/rs/db/amEmplDept/12345

List

This section describes how to use RESTful API functions to perform list operations.

Query list

GET rs/db/<TableName>?fields=<FieldNames>&filter=<Filter>&orderby=<FieldNames>&limit=<int>&offset=<int>

Response code: 200 / 404 if not found

Response body: A JSON-formatted list.

A sample is provided as follows.

URL:

http://localhost:8080/AssetManagerWebService/rs/db/amEmplDept?fields=CostCenter,CostCenter.Title,Absences,name,bCreatedOnTheFly&filter=pk>1&offest=20&limit=10&orderby=name

Content-Type: application/json

Response:

```
{
    "count": 127,                                          //Total number
    "entities": [
       {
            "Name": "Administration",                      //Normal field
            "CostCenter.Title": "Corporate services",      //1:1 field
            "CostCenter": {                                //1:1 link
                "ref-link": "db/amCostCenter/120711",
                "self": "Corporate services"
            },
            "bCreatedOnTheFly": {                          //System Itemized Lis
t
                "0": "No"
            },
            "MrMrs": "Mrs.",                               //Customized Itemized
List
            "self": "Administration, ",                    //Display name
            "Absences": {                                  //1:N link
                "ref-link": "db/amEmplDept/121238/Absences"
            },
```

```
            "ref-link": "db/amEmplDept/121238"                //Detail link
        },...
```

Query 1-N resources

GET rs/db/<TableName>/<PK>/<LinkName>

Response code: 200 / 404 if not found

Response body: A JSON formatted list.

A sample is provided as follows.

URL: http://localhost:8080/AssetManagerWebService/rs/db/amEmplDept/121241/Absences

Content-Type: application/json

Response:


```
{
    "count": 1,
    "entities": [
        {
            "dtLastModif": 1410214363000,
            "bArchived": {
                "1": "Yes"
            },
            "dtBack": 1402768800000,
            "self": "Listel, Susan / Annual meeting",
            "seValidated": {
                "1": "Validated"
            },
            "fDays": 1,
            "Field1": "Field1",
            "Nature": "Annual meeting",
            "dtValidation": 1389808800000,
            "AccountingType": "Annual convention",
            "PhoneContact": "06.12.11.81",
            "dtOut": 1402736400000,
            "Reason": "Annual meeting",
            "ref-link": "db/amAbsence/121850"
        }
    ]
}
```

Query URL parameters

Consider the following query URL pattern.

GET rs/db/<TableName>?fields=<FieldNames>&filter=<Filter>&orderby=<FieldNames>&limit=<int>&offset=<int>

Parameters:

- **<TableName>**: The Asset Manager database table name.
- **<FieldNames>**: A string separated by commas. For example, for the [amEmplDept] table, the string may contain:
  - Normal field: Database columns in the table of {tableName}. For example, "fields=Name,Title".
  - Calculated field: Support. For example, "fields=cf_seLoginClass".
  - 1-1 Link: The link's name. For example, "fields=CostCenter".
  - Sub normal field on 1-1 link. For example, "fields=CostCenter.code, CostCenter.Company.EMail".
  - Sub 1-1 link on 1-1 link. For example, "fields=CostCenter.Company".
  - All normal fields in {tableName} are returned if the "fields" parameter is not provided.
  - Response code 400 is returned if any invalid filed name is provided. For example, "fields=name,<text>".
  - Case insensitive. For example, "fields=name" equals to "fields=NAME".
  - Ignore spaces. For example, "fields=name , title " equals to "fields=name,title".

  Note: If you enter any of the following fields, the query result does not return the entity associated with the link.
  - 1-N Link
  - Sub normal field on 1-N Link
- **<Filter>**: The condition to filter the resource. Currently, it is a raw AQL WHERE clause.
- **<FieldNames>**: A string that is separated by commas. It contains:

- Normal field: Database columns in the table of {tableName}. For example, "orderby=name asc,fax desc".
- Sub normal field on 1-1 link. For example, "orderby=costcenter.code".
- The records are ordered by primary keys in ascending order if "orderby" is not provided.

Data types

The Asset Manager 9.50 RESTful APIs support the following data types.

- String

  Request: db/amEmplDept?fields=name

  Response: {"name": "Admin"}

- Integer

  Request: db/amAsset?fields=sMaxCnxCount

  Response: {"sMaxCnxCount": 10}

- Double

  Request: db/amAsset?fields=mMarketValRef1

  Response: {"mMarketValRef1": 1999.00}

- Date

  Request: db/amEmplDept?fields=dLeave

  Response: {"dLeave": 1423032441000} (Number of milliseconds since January 1, 1970, 00:00:00 GMT)

- DateTime

  Request: db/amEmplDept?fields=dtLastModif

  Response: {"dtLastModif": 1423032441000} (Number of milliseconds since January 1, 1970, 00:00:00 GMT)

- Long Text

  Request: db/amComment?fields=memComment

  Response: {"memComment": "This workflow deletes workflow instances that are completed since amWfScheme:fv_AutoCleaningDelay"}

- System Itemized List

  Request: db/amEmplDept?fields=bIsRCManager

Response: {"bIsRCManager": { "0": "No" }} (key : value pair provided)

- Customized Itemized List

  Request: db/amEmplDept?fields=MrMrs

  Response: {"MrMrs": "Mr"}

- Blob content

  Request: db/amImage/20425?fields=blbdata

  Response: binary content in response body

  Note: The blob query is supported because the value of the "fields" parameter may have only one value, for example, "fields=blbdata". If the "fields" parameter has more than one value, for example, "fields=blbData,name", the response body will be a JSON string with blob link:

  ```
  {
      "blbData": "db/amImage/20425?fields=blbData", //Blob link is provided
      "Name": "sysCoreRemoteControl.bmp",
      …
  }
  ```

- 1-1 link

  Request: db/amEmplDept?fields=CostCenter

  Response: {"CostCenter": { "ref-link": "db/amCostCenter/120703", "self": "Common Line" }}

- 1-N link

  Request: db/amEmplDept?fields=Absences

  Response: {"Absences": { "ref-link": "db/amEmplDept/240/Absences"}}

Appendix. How to implement life cycle for work order with RESTful API

In the example provided in this chapter, two users will log on to Asset Manager and perform CRUD operations with RESTful API.

The sample code is written in Java. We use Jersey client and JAX-RS as 3rd party libraries as HTTP client dependency in the following sample code. You can also use other HTTP client dependencies.

1. Configure for authorization.

```
import com.fasterxml.jackson.jaxrs.json.JacksonJaxbJsonProvider;
import org.glassfish.jersey.client.ClientConfig;
import org.glassfish.jersey.client.ClientProperties;
import org.glassfish.jersey.client.authentication.HttpAuthenticationFeature;

public class BeforeRequest {
    public ClientConfig config(String userName, String password) {

        //BASIC authenticate configuration
        HttpAuthenticationFeature feature = HttpAuthenticationFeature.basicBuil
der()
                .credentials(userName, password)
                .build();

        ClientConfig clientConfig = new ClientConfig();
        clientConfig.register(feature);
        clientConfig.property(ClientProperties.FEATURE_AUTO_DISCOVERY_DISABLE,
true);
        clientConfig.register(JacksonJaxbJsonProvider.class);

        return clientConfig;
    }
}
```

2. Log on to Asset Manager as an IT manager and then create a new work order for one technician.

```
public void create() {
        BeforeRequest beforeRequest = new BeforeRequest();
        clientConfig = beforeRequest.config("Username", "Password");
         Map<String, String> map = new HashMap<>();
         map .put("WONo", "RestTestWO1");
         map .put("ReqLine.lLineNumber", "request line number"); // Please type
a valid request line here
         map .put("Technician.BarCode", "User barcode"); // Please type a valid
```

```
user barcode here.

          Response res = ClientBuilder.newClient(clientConfig)
                    .target("http://localhost:8080/AssetManagerWebService/rs/db
/amWorkOrder")
                    .request(MediaType.APPLICATION_JSON_TYPE)
                    .post(Entity.entity(map, MediaType.APPLICATION_JSON_TYPE));

           String recordId = res.readEntity(String.class); // The main id of ne
w record

       }
```

3. Log on as the technician and then query the work order created in step 2.

```
public void read() {
        BeforeRequest beforeRequest = new BeforeRequest();
        clientConfig = beforeRequest.config("Username", "Password");

        Map queryEntity = ClientBuilder.newClient(clientConfig)
                .target("http://localhost:8080/AssetManagerWebService/rs/db/amW
orkOrder")
                .queryParam("fields", "lWorkOrderId,lReqLineId,ReqLine.lModelI
d," +
                        "ReqLine.Model.Nature.seBasis,ReqLine.Model.Nature.Over
flowTbl")
                .queryParam("filter", "Technician.barCode  = '" + userName + "'
AND seStatus = 0" )
                .request(MediaType.APPLICATION_JSON_TYPE)
                .get(Map.class);

       List<Map<String, Object>>  workOrders = (List<Map<String, Object>>) quer
yEntity.get("entities");
    }
```

4. Update the work orders.

```
public void update(String lWorkOrderId) {
        BeforeRequest beforeRequest = new BeforeRequest();
        clientConfig = beforeRequest.config("Username", "Password");
        Map<String, String> map = new HashMap<>();
         map.put("dtNotif", "1234566783");  // Please convert date to milliseco
nd
         map.put("dtActualFixStart", "1334534566");
         map.put("dtActualFixed", "3435665777");

         Response res = ClientBuilder.newClient(clientConfig)
                   .target
```

```
("http://localhost:8080/AssetManagerWebService/rs/db/amWorkOrder/" + lWorkOrder
Id)
                    .request(MediaType.APPLICATION_JSON_TYPE)
                    .post(Entity.entity(map, MediaType.APPLICATION_JSON_TYPE));
         // res.getStatus() == 200
     }
```

5. Delete the work orders.

```
public void delete(String lWorkOrderId) {
        BeforeRequest beforeRequest = new BeforeRequest();
        clientConfig = beforeRequest.config("Username", "Password");
        Response res = ClientBuilder.newClient(clientConfig)
                    .target("http://localhost:8080/AssetManagerWebService/rs/db
/amWorkOrder" )
                    .path(lWorkOrderId).request().delete();
          // res.getStatus() = 200
     }
```

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Asset Manager 9.50 RESTful API (Asset Manager 9.50)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to ovdoc-ITSM@hp.com.

We appreciate your feedback!