X4 REST API

Created by Dimitar Hristov (February 2017)
Last modified: April 2018

The eMaint RESTFul API web service

Table of Contents:

Introduction and General Information

Third party applications are now able to communicate directly with X4 through the new REST API supplied. This gives us more flexibility against using SOAP and is originally created for FCCM integration.

Accessibility

The API is accessible through the following general URL

http://<emaint server>/wc.dll?x3~api~&q=<WEB_METHOD>

X4 will expose a limited web methods available within this API.

Available servers

testx32.emaint.com - test server

x41.emaint.com - production server
x42.emaint.com - production server
x43.emaint.com - production server
x44.emaint.com - production server
x45.emaint.com - production server
x46.emaint.com - production server
x47.emaint.com - production server
x48.emaint.com - production server

General usage

User needs to obtain X4 token and use the token to any subsequent calls. Once the token is generated it stays valid for the next 10 years. Any information send as part of the authentication process must present in the header of the request message.Please refer to GetLoginToken method about how to obtain a valid token.

Information:
All webmethods can now accept additional header parameter **DataFormat** with acceptable values of JSON or XML. If passed, the response would be formatted against the requested data format.

Web Methods

:: GetLoginToken

Method:

GET

URL:

http://<emaint server>/wc.dll?x3~api~&q=GetLoginToken

Header:

- **XT-UserAgent** - free text, the login token generated would be mapped to the user agent specified; same user agent id must present in subsequent calls
- **Username** - X4 user name
- **Password** - X4 password (plain text)

Request:

```
POST /wc.dll?x3~api~&q=GetLoginToken HTTP/1.1
Host: testx32.emaint.com
XT-UserAgent: USERA
Username: EMAINTTEST
Password: EMAINTTEST
Cache-Control: no-cache
Content-Type: multipart/form-data;
```

Response:

```
{
  "valid": "true",
  "message": "",
  "token": "5DFF86F8-FC84-4245-8C59-46A2E555D987"
  "serverUrl": "http://x41.emaint.com/
}
```

Or

```
{
  "valid": "false",
  "message": "Authentication failure"
}
```

GetLoginToken may be called from any server in the same environment, but all other methods must be called from the server provided in the “serverUrl” value.

:: FlukeNotification

Method:
POST
URL:
http://<emaint server>/wc.dll?x3~api~&q=FlukeNotification
Header:

- **XT-UserAgent** - free text, the login token generated would be mapped to the user agent specified; same user agent id must present in subsequent calls
- **Authenticate** - X4 login token as received from GetLoginToken method

Response:
When asset does not exist:

```
{
  "valid": "false",
  "message": "Asset CE501ZZZZ does not exists in database. Notification 
ignored.",
  "workOrderCreated": false,
  "workOrderNumber": null
}
```

When work order created:

```
{
  "valid": "true",
  "message": "Work order #35847 for asset CE502 created successfully.",
  "workOrderCreated": true,
  "workOrderNumber": 35847
}
```

When work order already created for this alert and is still open in X4:

```
{
  "valid": "true",
  "message": "Work order #35846 for asset CE501 already exist. No work
order created.",
  "workOrderCreated": false,
  "workOrderNumber": 35846
}
```

Request:

```
POST /wc.dll?x3~api~&amp;q=FlukeNotification HTTP/1.1
Host: testx32.emaint.com
XT-UserAgent: USERA
Authenticate: 5DFF86F8-FC84-4245-8C59-46A2E555D987
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded

{
 "high": 100.0,
 "monitoringPointId": "SE108",
 "triggeredValue": 110.01,
 "low": 0.0,
 "assetId": "SE108",
 "um": "V DC",
 "notificationId": "cc4e5ed5-5700-4eec-b268-558113f3f736",
 "alarmMessage": "ACTUAL: 110.01 VDC LIMIT: \u003e 100 VDC ASSET: T1 -
Capper",
 "createdOn": 1487688849830
}
```

:: Assets

Method:

GET

URL:

http://<emaint server>/wc.dll?x3~api~&q=Assets

Descriptions:

Returns a XML message with information about the assets marked as AMR assets in X4 (COMPINFO.AMRASSET=.T.). If AMRASSET field does not exist, all assets are returned. (NOTE: This is no longer used with AMR!)

Response type:

XML

Notes:

Schad AMR v1 specific method, can be accessed using XT-UserAgetnt=SCHAD_AMR only.

Header:

- **XT-UserAgent** – free text, the login token generated would be mapped to the user agent specified; same user agent id must present in subsequent calls; this method works with XT-UserAgent=SCHAD_AMR only
- **Authenticate** – X4 login token as received from GetLoginToken method

Request:

```
GET /wc.dll?x3~api~&amp;q=Assets HTTP/1.1
Host: testx32.emaint.com
XT-UserAgent: USERA
Authenticate: 32E3431A-20B5-4FFB-B17D-A0D005822075
DataFormat: JSON
Cache-Control: no-cache
```

Response:

When one or more assets returned (XML):

```
<RESPONSE>
     <VALID>true</VALID>
     <MESSAGE></MESSAGE>,
     <DATA>
        <ASSETS>
           <ASSET>
```

```
                <assetid>TTTXXX</assetid>
                <admindesc>PIV 22</admindesc>
                <objectstatusid>ACTIVE</objectstatusid>
           </ASSET>
           <ASSET>
                <assetid>FC3000TEST</assetid>
                <admindesc>X4 - FC 3000 Measurement Test</admindesc>
                <objectstatusid>ACTIVE</objectstatusid>
           </ASSET>
         </ASSETS>
      </DATA>
</RESPONSE>
```

In case of authentication failure error:

```
<RESPONSE>
     <VALID>false</VALID>
     <MESSAGE>Authentication failure.</MESSAGE>
</RESPONSE>
```

:: MonitoringPoints

Method:
GET

URL
http://<emaint server>/wc.dll?x3~api~&q=MonitoringPoints&assetId=<ASSETID>

Descriptions:
Returns a message with information about the monitoring points assigned to the provided <ASSETID> in X4.

Response type:
XML/JSON

Notes:
Schad AMR v1 specific method, can be accessed using XT-UserAgetnt=SCHAD_AMR only.

Header:

- **XT-UserAgent** - free text, the login token generated would be mapped to the user agent specified; same user agent id must present in subsequent calls; this method works with XT-UserAgent=SCHAD_AMR only
- **Authenticate** - X4 login token as received from GetLoginToken method

Request:

```
GET /wc.dll?x3~api~&amp;q=MonitoringPoints&amp;assetId=FLUKECONNECTTESTASSET
HTTP/1.1
Host: testx32.emaint.com
XT-UserAgent: USERA
Authenticate: 32E3431A-20B5-4FFB-B17D-A0D005822075
DataFormat: JSON
Cache-Control: no-cache
```

Response (JSON):

```
{
    "valid": "true",
    "message": "",
    "data": [
    {
         "monit_id": "_4YO0B23UH",
         "monit_type": "Vertical",
         "mon_units": "mm/s",
         "nlower": 0,
         "nupper": 0
    },
    {
         "monit_id": "_4YO0B23UG",
         "monit_type": "Horizontal",
         "mon_units": "mm/s",
         "nlower": 0,
         "nupper": 0
    },
    {
         "monit_id": "_4YO0B23UF",
         "monit_type": "Axial",
         "mon_units": "mm/s",
         "nlower": 0,
         "nupper": 0
```

```
      }
      ]
}
```

:: Reading

Method:

PUT

URL:

http://<emaint server>/wc.dll?x3~api~&q=Reading

Descriptions:

Collects one or more condition monitoring readings from AMR tool and stores the readings in MON_READ table against the provided asset id.

Request type:

JSON raw data is expected. JSON array is accepted if submitted more than one reading. In case only 1 reading is submitted, system can also accept single JSON object

Response type:

XML/JSON

Notes:

Schad AMR v1 specific method, can be accessed using XT-UserAgetnt=SCHAD_AMR only.

Header:

- **XT-UserAgent** – free text, the login token generated would be mapped to the user agent specified; same user agent id must present in subsequent calls; this method works with XT-UserAgent=SCHAD_AMR only
- **Authenticate** – X4 login token as received from GetLoginToken method

Request:

```
PUT /wc.dll?x3~api~&amp;q=Reading HTTP/1.1
Host: testx32.emaint.com
XT-UserAgent: USERA
Authenticate: 32E3431A-20B5-4FFB-B17D-A0D005822075
DataFormat: JSON
```

```
Cache-Control: no-cache
[{
    "siteId": "",
    "assetId": "FLUKECONNECTTESTASSET",
    "meterId": "_4YO0B23UH",
    "meterType": "mm/s",
    "changeBy": "AMR",
    "changeDate": "2017-05-03",
    "newReadingDate": "2017-07-13",
    "newReading": 33
},
{...}]
```

Response (JSON):
When reading is validated and saved successfully.

```
{
     "valid": "true",
     "message": "Reading saved."
}
```

Response (JSON):
When validation failed.

```
{
     "valid": "false",
     "message": "ERROR: The reading of <b>30</b> is less than  the last
recorded reading of <b>33</b><BR><BR><font color=red><b>Your entry will
not be saved!</b></font><br><br>"
}
```

:: GetAnyData

Method:

POST

URL:

`http://<emaint server>/wc.dll?x3~api~&q=GetAnyData`

Descriptions:

Return the result of a query based on parameters passed which defines:

- Table name
- Columns listed
- Filter applied
- Sorted columns and directions
- Page number
- Page size
- Object Structure ID

Request type:

JSON raw data is expected with following format:

```
{
 "table": "<TABLE NAME>",
 "columns": "<COMMA SEPARATED LIST OF COLUMNS>",
 "filter": <FILTER OBJECT IN KENDO FORMAT>,
 "sortBy": <ARRAY OF OBJECTS IN SPECIFIC FORMAT>,
 "pageNumber": <PAGE NUMBER REQUESTED>,
 "pageSize": <NUMBER OF RECORDS PER PAGE>
}
```

Response type:

XML/JSON

Notes:

Data restrictions may apply if the user doesn't have access to particular table data.

sortBy - is an array of obects. Each object must have the following format:
{"field":<FIELD NAME>, "dir": "ASC|DESC"}

Filter - is an object that defines the filter. Any complexity can be used. The format of the filter is as following:
{"logic":"AND|OR","filters":<FILTER>}

<FILTER> can define:

- Array of field objects in the following format:

{"field":"<TABLE FIELD NAME>","operator":"<OPERATOR>","value":"<VALUE>"}

- Inner filter configuration in the following format:

{"logic":"AND|OR","filters":<FILTER>}

The following simple example defines a filter of type:
(A AND B AND C)

```
{
    "logic": "and",
    "filters": [{
        "field": "work__compid",
        "operator": "startswith",
        "value": "MX"
    }, {
        "field": "work__line_no",
        "operator": "startswith",
        "value": "PLR"
    }, {
        "field": "work__date_wo",
        "operator": "eq",
        "value": "2016-01-26T00:00:00.000Z"
    }]
}
```

The above translates to:
(Asset ID **Starts with** **"MX"** And Line Number **Starts with** **"PLR"** AND Work Order Date **Equals to** **"26 Jan 2016"**)

The following complex example defines a filter of type:
(A AND B) OR (C AND D):

```
{
    "logic": "OR",
    "filters": [{
        "logic": "AND",
        "filters": [{
            "field": "work__aprv_pri",
            "operator": "eq",
            "value": "1"
        }, {
            "field": "work__action",
            "operator": "eq",
            "entryas": "value",
            "value": "TEST"
        }]
    }, {
        "logic": "AND",
        "filters": [{
            "field": "work__aprv_pri",
            "operator": "eq",
            "value": "2"
        }, {
            "field": "work__billable",
            "operator": "eq",
            "value": "true"
        }]
    }]
}
```

The above translates to:

(Approval Priority **Is equal to** **1** And Action **Is equal to** **TEST**) Or (Approval Priority **Is equal to** **2** And Billable? **Is equal to** **true**)

NEW! As of 20.02.2018, system now accepts simplified method to describe basic filters such as:

```
{
    "field": "COMPID",
    "operator": "eq",
    "value": "A1003"
}
```

The above translates to:
(Equipment ID **Is equal to** **A1003**)

List of available operators used in filter:

- eq - equals to
- neq - not equals to
- gte - greater than or equal to
- gt - greater than
- lte - less than or equal to
- lt - less than
- startswith - starts with
- notstartswith - does not start with
- endswith - ends with
- notendswith - does not end with
- contains - contains
- containany - contains any of the words in value list (space separated)
- containall - contains all of the words in value list (space separated)
- notcontains - does not contain
- isempty - is empty
- notisempty - is not empty
- isnull - is null

- notisnull - is not null
- between - is between "value" and "value2"
- notbetween - is not between "value" and "value2"

Date values are represented in ISO format.

Header:

- **XT-UserAgent** - free text, the login token generated would be mapped to the user agent specified; same user agent id must present in subsequent calls;
- **Authenticate** - X4 login token as received from GetLoginToken method

Request:

The following example will construct the following query:

SELECT COMPID,COMP_DESC FROM COMPINFO WHERE comp_desc like "%HVAC% ORDER BY COMP_DESC DESC

The output will contain a set of records defined by the paging options. In this case records number 31-35 will be returned in response.

```
POST /wc.dll?x3~api~&amp;q=GetAnyData HTTP/1.1
Host: testx32.emaint.com
XT-UserAgent: USERA
Authenticate: 32E3431A-20B5-4FFB-B17D-A0D005822075
DataFormat: JSON
Cache-Control: no-cache

{
 "table": "COMPINFO",
 "columns": "COMPID,COMP_DESC",
 "filter":
{"logic":"and","filters":[{"field":"comp_desc","operator":"contains","value"
:"HVAC"}]},
 "sortBy": [{"field":"comp_desc", "dir":"desc"}],
 "pageNumber": 7,
 "pageSize": 5
}
```

Response (JSON):

```
{
    "valid": "true",
    "message": "",
    "data": [
        {
            "compid": "HV494",
            "comp_desc": "Y29 - HVAC"
        },
        {
            "compid": "HV493",
            "comp_desc": "Y27 - HVAC"
        },
        {
            "compid": "HV492",
            "comp_desc": "Y24 - HVAC"
        },
        {
            "compid": "HV491",
            "comp_desc": "Y22 - HVAC  Feeds 2nd floor Tea Mechanical
room"
        },
        {
            "compid": "HV490",
            "comp_desc": "Y21 - HVAC  Feeds 1st Floor East Side Distrib
Duct"
        }
    ]
}
```

:: Record

Method:

POST, PUT, DELETE

URL:

http://<emaint server>/wc.dll?x3~api~&q=Record

Descriptions:

Use this method to create, update or delete record from the account's database

- With method POST - updates existing record with the data provided in “payload” json object
- With method PUT - creates new record with the data provided in “payload” json object
- With method DELETE - deletes or restores existing record

Request type:

JSON raw data is expected with following format:

When POST web method used:

```
{
    "table": "WORK",
    "id": "34527",
    "payload": {
          "brief_desc" : "Test brief description",
          "assignid" : "Test Assign ID"
    }
}
```

When PUT web method used:

```
{
    "table": "WORK",
    "payload": {
           "wo" : "[AUTO]",
          "brief_desc" : "Test brief description",
          "assignid" : "Test Assign ID",
          "date_wo" : "2017-11-02T00:00:00.000",
          "standard" : 3.50
    }
}
```

When DELETE web method used:

```
{
    "table": "WORK",
    "id": "34527",
    "action”: "delete|restore"
}
```

Response type:

XML/JSON

Notes:

- This web method may update only one record at time.
- There are no restriction of the data submitted - there is no validation against correct master/detail data submitted
- There is no validation against applicable values that a drop-down field can take, thus you may submit a value that is not in the list of selectable values in X4
- Date/time values are expected in UTC format
- User may omit "action" when deleting record. Delete action is by default

:: GetObjectData

Method:

GET

URL:

http://<emaint server>/wc.dll?x3~api~&q=GetObjectData&TABLE={DATADICT.DD_DBF}&KEYVALUE={RECORD_ID}&OBJECTID={BOSMODEL.BOSID}

Descriptions:

Use this method to retrieve table record information in a complex XML or JSON format as defined in Object Structure record referred.

Request type:

Required information to be passed in Query String parameters:

- TABLE - table name as defined in DATADICT.DD_DBF ("D" record type)
- KEYVALUE - record identifier
- OBJECTID - object structure identifier (BOSMODEL.BOSID)

Response type:

XML/JSON

Response examples:

When JSON DataType requested:

```
{
    "valid": "true",
    "message": "",
    "data": {
        "WORK": {
            "WO": "742",
            "COMPID": "HL11150",
            "STATTYPE": "",
            "WORKSTATUS": "O",
            "BRIEF_DESC": "CONVEYOR BELT GENERAL INSPECTION",
            "COMPINFO": {
                "COMPID": "HL11150",
                "COMP_DESC": "Filler Empty Bottle Infeed Screw Conveyor"
            },
            "CHARGES": {
                "DATA": []
            },
            "WO_PROCS": {
                "DATA": [
                    {
                        "CUID": "_54D0I4U5K",
                        "PROC_SEQ": "1",
                        "TASK_NO": "A-1",
                        "TASK_DESC": "A-1 Air Dryer, Refrigerated or
Regenerative",
                        "COMPLETE": "",
                        "STANDARD": "0.00",
                        "WO": "742"
                    },
                    {
                        "CUID": "_54D0I5B9V",
                        "PROC_SEQ": "2",
                        "TASK_NO": "A-2",
                        "TASK_DESC": "A-2 Unitary, Heating and Cooling
Unit",
                        "COMPLETE": "",
                        "STANDARD": "0.00",
                        "WO": "742"
                    }
                ]
            }
        }
    }
}
```

    When XML DataType requested:

```
<RESPONSE>
    <VALID>true</VALID>
    <MESSAGE></MESSAGE>,
    <DATA>
        <WORK>
            <WO>742</WO>
            <COMPID>HL11150</COMPID>
            <STATTYPE></STATTYPE>
            <WORKSTATUS>O</WORKSTATUS>
            <BRIEF_DESC>CONVEYOR BELT GENERAL INSPECTION</BRIEF_DESC>
            <COMPINFO>
                <COMPID>HL11150</COMPID>
                <COMP_DESC>Filler Empty Bottle Infeed Screw
Conveyor</COMP_DESC>
            </COMPINFO>
            <CHARGES></CHARGES>
            <WO_PROCS>
                <DATA>
                    <CUID>_54D0I4U5K</CUID>
                    <PROC_SEQ>1</PROC_SEQ>
                    <TASK_NO>A-1</TASK_NO>
                    <TASK_DESC>A-1 Air Dryer, Refrigerated or
Regenerative</TASK_DESC>
                    <COMPLETE></COMPLETE>
                    <STANDARD>0.00</STANDARD>
                    <WO>742</WO>
                </DATA>
                <DATA>
                    <CUID>_54D0I5B9V</CUID>
                    <PROC_SEQ>2</PROC_SEQ>
                    <TASK_NO>A-2</TASK_NO>
                    <TASK_DESC>A-2 Unitary, Heating and Cooling
Unit</TASK_DESC>
                    <COMPLETE></COMPLETE>
                    <STANDARD>0.00</STANDARD>
                    <WO>742</WO>
                </DATA>
            </WO_PROCS>
        </WORK>
    </DATA>
```

```
</RESPONSE>
```

The following Postman link contains a test request to GetObjectData API method:
https://www.getpostman.com/collections/c9ae6e83a9ad3f88c87e

Notes:

- This web method may return information for single master record.
- The method can return the response in both formats XML or JSON
- Date/time values are expected in UTC format

:: CloseWorkOrder

Method:

POST

URL:

http://<emaint server>/wc.dll?x3~api~&q=CloseWorkOrder

Descriptions:

Use this method to close out a work order

Request type:

JSON raw data is expected with following format:

```
{
    "wo": "12345",
    "date_cmpl": "2018-04-21T00.00.00.000Z"
}
```

Response type:

XML/JSON

Response examples:

When JSON DataType requested:

```
{
    "valid": "true",
    "message": "Work order successfully closed"
}
```

When XML DataType requested:

```
<RESPONSE>
    <VALID>true</VALID>
    <MESSAGE>Work order successfully closed</MESSAGE>
</RESPONSE>
```

The following Postman link contains a test request to CloseWorkOrder API method:
https://www.getpostman.com/collections/c9ae6e83a9ad3f88c87e

:: SignOn

Method:
POST

URL:
http://<emaint server>/wc.dll?x3~api~&q=SignOn

Descriptions:
Use this method to sign on to a work order

Request type:

JSON raw data is expected with following format:

```
{
    "wo": "12345",
    "signOnDate": "2018-04-21T00.00.00.000Z",
    "contactId": "BWDEMO"
}
```

Response type:
XML/JSON

Response examples:
When JSON DataType requested:

```
{
    "valid": "true",
    "message": "Successfully signed on to work order #254"
}
```

When XML DataType requested:

```
<RESPONSE>
    <VALID>true</VALID>
    <MESSAGE>Successfully signed on to work order #254</MESSAGE>
</RESPONSE>
```

Notes:

- When signOnDate is omitted, current server date/time is used.
- The sign on user is determined by the contactId parameter submitted in the payload
- Sign on rules are evaluated before allowing user to sign on. There might be Clock In/Out restrictions, or restrictions to sign on to multiple work orders at once.
- SignOnDate cannot be a future date
- The work order must exist and not deleted

:: SignOff

Method:
POST

URL:
http://<emaint server>/wc.dll?x3~api~&q=SignOff

Descriptions:

Use this method to sign off to a work order. You can use this method to pause working on a work order as well

Request type:

JSON raw data is expected with following format:

```
{
    "wo": "12345",
    "signOffDate": "2018-04-21T00.00.00.000Z",
    "contactId": "BWDEMO",
    "completed": true
}
```

Response type:

XML/JSON

Response examples:

When JSON DataType requested:

```
{
    "valid": "true",
    "message": "Successfully signed off to work order #254"
}
```

When XML DataType requested:

```
<RESPONSE>
    <VALID>true</VALID>
    <MESSAGE>Successfully signed off to work order #254</MESSAGE>
</RESPONSE>
```

Notes:

- When signOffDate is omitted, current server date/time is used.
- The sign on user is determined by the contactId parameter submitted in the payload
- Sign on rules are evaluated before allowing user to sign on. There might be Clock In/Out restrictions, or restrictions to sign on to multiple work orders at once.

- SignOffDate cannot be a future date
- The work order must exist and not deleted
- Completed parameter is not required
- If Complete equals to “false” - user will “pause” working on the selected work order. To re-start, use SignOn API again

:: ApproveRequest

Method:
POST

URL:
http://<emaint server>/wc.dll?x3~api~&q=ApproveRequest

Descriptions:
Use this method to approve work request. Approved work requests generate work order automatically.

Request type:

JSON raw data is expected with following format:

```
{
    "requestNo": "12345",
    "contactId": "EMAINT"
}
```

Response type:
XML/JSON

Response examples:
When JSON DataType requested:

```
{
    "valid": "true",
    "message": "Work Request approved on 08/28/2018 : 10:29:09 AM By EMAINT.
Request Number 01386 became WO No. 34620",
    "wo": 34620
}
```

Notes:

- When contactId is omitted, current logged in user used.
- The work request must exist and not deleted

:: RejectRequest

Method:

POST

URL:

http://<emaint server>/wc.dll?x3~api~&q=RejectRequest

Descriptions:

Use this method to reject work request.

Request type:

JSON raw data is expected with following format:

```
{
    "requestNo": "12345",
    "contactId": "EMAINT",
    "reason": "We cannot approve this request due to technical reasons",
}
```

Response type:

XML/JSON

Response examples:

When JSON DataType requested:

```
{
    "valid": "true",
    "message": "Work Request Number 01351 has been REJECTED."
}
```

Notes:

- When contactId is omitted, current logged in user used.

- The work request must exist and not deleted
- Rejection reason must be supplied

:: UploadDocument

Method:

PUT

URL:

http://<emaint server>/wc.dll?x3~api~&q=UploadDocument

Descriptions:

Use this method to upload document or image file.

Request type:

JSON raw data is expected with following format:

```
{
    "filename": "any file.jpg",
    "description": "file description",
    "folder": "MYFOLDER",
    "data":
"iVBORw0KGgoAAAANSUhEUgAAAuoAAAGmCAYAAAAqFaEsAAAAAXNSR0IArs4c6QAAAARnQU1BAAC
xjwv8YQUA...",
}
```

Response type:

XML/JSON

Response examples:

When JSON DataType requested:

```
{
    "valid": "true",
    "message": "File uploaded successfully.",
    "id": "_5AJ0QW63L"
}
```

Notes:

- Description is optionable
- Folder is optionable. If missing, files are uploaded to MOBILE folder under the root folder
- Image files are compressed before saved in document storage
- The size of of the uploaded image or document is limited to 16MB

:: DownloadDocument

Method:

GET

URL:

http://<emaint server>/wc.dll?x3~api~&q=DownloadDocument&id=<<ID>>

Descriptions:

Use this method to download existing document or image file.

Request type:

Request:

```
POST /wc.dll?x3~api~&q=DownloadDocument&id=_5AJ0QW63L HTTP/1.1
Host: testx32.emaint.com
XT-UserAgent: USERA
Authenticate: B5F7A361-Z66F-464B-AEEE-1A087CA63324
Cache-Control: no-cache
```

Response:

```
{
  "valid": "true",
  "message": "",
  "data": "/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAgGBgcGBQgH…"
}
```

:: AdjustInventory

Method:

POST

URL:

http://<emaint server>/wc.dll?x3~api~&q=AdjustInventory

Descriptions:

Use this method to submit adjustments on inventory item on hand quantities or transfer quantities from one location to another. There are two types of allowed adjustments:

- ONHAND - new on hand is directly specified
- ADJUSTMENT - on hand value is specified as adjustment to current on hand value (+/-)

Adjustment document type is specified in "document" field of the payload. All available options are listed below:

- PHYSCOUNT (Physical Count) [default]
- CYCLECOUNT (Cycle Count)
- XFERIN Transfer (In)
- XFEROUT (Transfer (Out)
- XFERLOC (Transfer (to Location)
- OVERAGE (Overage)
- SHORTAGE (Shortage)
- PORETURN   (P.O. Return)
- PILFERAGE (Pilgerage)
- TOOL_IN (Tool Crib In)
- TOOL_OUT (Tool Crib Out)
- CONSUMABLE (Production usage)
- OTHER (Other)

Note: For a complete list of available adjustment types, check out in Account Settings -> Inventory Control Settings section

Other important payload fields:
"Qtyadj" - adjustment quantity value
"Invloc" - inventory location

Request type:

JSON raw data is expected with following format:

```
{
      "type":"ONHAND|ADJUSTMENT",
      "payload": {
            "item":"FILTER01",
            "invloc":"MAIN",
            "document":"PHYSCOUNT",
            "qtyadj":4
      }
}
```

Response type:
XML/JSON

Response examples:
When JSON DataType requested:

```
 {
     "valid": "true",
     "message": "Your transaction was saved."
 }
```

Notes:

- Qtyadj is mandatory field
- In case adjustment “type” is not specified system uses “ADJUSTMENT” type by default
- In case “invloc” (inventory location) is not specified, MAIN is used by default
- Adjustment document type is PHYSCOUNT and is used by default if not specified

:: GetPermissions

Method:
GET

URL:
`http://<emaint server>/wc.dll?x3~api~&q=GetPermissions&Userid=<<UserName>>`

Descriptions:
Use this method to extract the set of applied user permissions.

Response type:
XML/JSON

Response examples:
When JSON DataType requested:

```
{
    "valid": "true",
    "message": "Your transaction was saved."
}
```

Notes:

- Administrator user have all permissions applied
- Requester users have a set of pre-defined permissions “hard-coded” in X4.
- The permissions of the standard user may be applied by one or more user roles.
- If you don’t specify &UserId parameter, the permissions of the current user is returned.

:: SavePOReceipt

Method:
POST

URL:
`http://<emaint server>/wc.dll?x3~api~&q=SavePOReceipt`

Descriptions:
Use this method to submit receive PO transaction quantities and/or close transaction lines. There are two transaction types supported
The “totalrec” type will be used to do a comparison with the value sent to the API and the value currently in the POTRAN.RECQTY field. If the number is less than or the same value as the existing RECQTY field, the value will be rejected. If the value is greater, the amount that is

receive will be based on the totalrec - RECQTY on the record.

- TOTALREC - used to do a comparisson with the value sent to the API and the value currently in POTRAN.RECQTY field. If the qty submitted is less than or the same value as the existing RECQTY field, the value will be rejected. If the value is greater, the amount that is receive will be based on the QTY (submitted) - RECQTY on the record.
- QTYREC - is used to post how many items have been received since the last transaction. Regardless of what type is in the POTRAN.RECQTY field the qty produced is the amount that will be received and added to the existing RECQTY value.

Request type:

JSON raw data is expected with following format:

```
{
     "purno": "00015",
     "type": "TOTALREC | QTYREC (default)",
     "payload": [
          {
               "lineid": "_5AM0XBEG8",
               "qty": 1
          },
          {
               "lineno": 2,
               "qty": 2
          },
          {
               "item": "2005",
               "qty": 2,
                "close": true | false (default)
          }
     ]
}
```

Response type:

XML/JSON

Response examples:

When JSON DataType requested:

```
{
    "valid": "false",
    "message": "One or more transactions have failed.",
    "details": [
        {
            "valid": false,
            "item": "1001",
            "lineid": "_5AM0XBEG8",
            "message": "Error! - Qty received greater than ordered"
        },
        {
            "valid": false,
            "item": "1000",
            "lineid": "_5AM0XBEG9",
            "message": "Error! - Qty received greater than ordered"
        },
        {
            "valid": true,
            "item": "2005",
            "lineid": "_5AM0XBEG7",
            "message": "Transaction passed."
        }
    ]
}
```

Notes:

- An array of transactions is expected
- All transactions passed must belong to same PO number
- Transactions can be found by LINEID, LINENO or ITEM.
- Each transaction is processed separately. Therefore, some may succeed but others may fail
- The status of each transaction is displayed in response in "details" object (as an array of objects)