FocusOPEN
DIGITAL ASSET MANAGER

FocusOPEN Scripting & API Overview

1 February 2012
Revision: 1

Notes
The information applies to the 3.3.9.5 release and above. Specific reference is made to 3.4.2 functionality in the pre-written API calls.

Contents

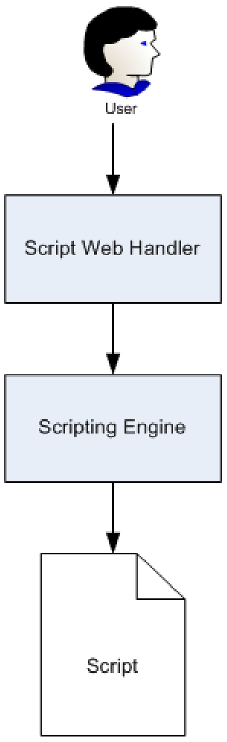
Overview

The FocusOPEN Scripting & API facility enables custom features and control of the FocusOPEN system without directly modifying the core application and making it incompatible with future updates. The API is a relatively recent addition to FocusOPEN and new functionality is being added with each new release.

Scripting & API Architecture

There are two elements to the FocusOPEN extensibility architecture: the scripting engine and the REST API. The Scripting Engine allows custom server-side JavaScript scripts to be developed and called. The REST API is authored using the Scripting Engine and mediates Calls to scripts. FocusOPEN uses the Jint JavaScript scripting engine to provide a server-side JavaScript scripting facility that can be used to extend FocusOPEN.

The chain of execution is:



The preferred method of calling scripts is using the REST API but it is technically possible to initiate a script.

REST API

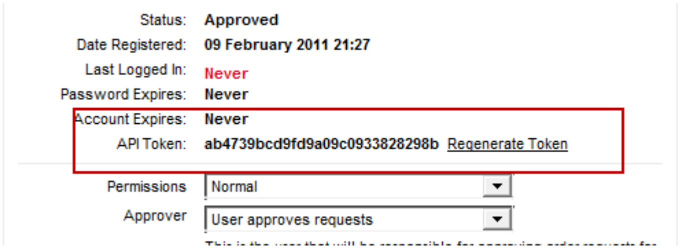
The REST API uses JSON as the default protocol for exchanging data. Most current Calls are made using http GET requests (with POST planned for some up-coming API Calls).

Authentication Theory

To access API methods, the client must login using a persistent User API Token linked to a FocusOPEN user account. Upon successfully logging in, a session token is returned. The session token is used to verify that the user is logged in. The user must pass a session token for every operation.

User API Token
Every user created in FocusOPEN has an API token created as soon as their account is added or they validated following registration. An API token is used to avoid passing around their actual password (or the hashed equivalent of it).

In the user management page, the user's User API Token is displayed and a link next to it: regenerate token. If this is clicked, a new token is created. This feature is to allow the User API Token to be changed if necessary.



Status:	Approved
Date Registered:	09 February 2011 21:27
Last Logged In:	Never
Password Expires:	Never
Account Expires:	Never
API Token:	ab4739bcd9fd9a09c0933828298b Regenerate Token
Permissions	Normal
Approver	User approves requests

Obtaining a Session Token
The process of obtaining a session token is to make an API call to login. If the login is successful, a session token is returned, this is a random identifier which will only persist for the duration of the session. The session details are recorded in two fields in the User table:

- Session API Token
- Last API Operation Date/Time

The last API operation date/time is updated each time an API operation is successful (including initially logging in). If the user logs out (either via the API or using the web interface) the Session API Token is reset to null.

Validating a Login Using the Session API Token
When an API operation is called this process occurs:

1. The session token is checked against the user table to see if it corresponds to a recognised user.
2. If the session token is not found, the operation is rejected.
3. If the session token matches but the current date/time is later than the last successful operation plus the default session length of the web application then the operation fails because the session has expired.

Assuming all conditions are satisfied, the API call is permitted.

FocusOPEN Web Application Automatic Client-Side API Session Token Binding

The API can be used by client-side scripts contained on custom home pages or your own preview plug-ins from within the FocusOPEN application itself. There is an automatic client-side binding on every FocusOPEN page so if calling API functions, it is not necessary to login twice and/or expose the User API token (which could be accessed by other visitors viewing the source of the page).

The JavaScript function template is as follows:

```
function getAPISessionToken(){
        return "[session token]";
}
```

Where [session token] = the session token received. The session token is transient so must be checked prior to each operation. Any client-side code built into FocusOPEN can then use this automatically.

Authentication Example

The following shows an example of how authentication is carried out:

[URL]/scripts.REST.users.login.ashx?userAPIToken=[User API Token]

Example:

[URL] = http://localhost/focusopen/
[User API Token] = fffd50fe65fa9feeb1e16c78e00aa4b2

The Call would be:

http://localhost/focusopen/scripts.REST.users.login.ashx?userAPIToken=fffd50fe65fa9feeb1e16c78e00aa4b2

If the API token is correct, FocusOPEN will return:

{"call":"login","data":{"sessiontoken":"[Session Token]"},"datime":"[JSON Format Time]","errors":{"code":0,"description":"Successful"},"requestid":"[Request Hash]","responseid":"[Response Hash]"}

Example:

{"call":"login","data":{"sessiontoken":"79e46e7c-8105-4161-bbe6-9642027051b4"},"datime":"\/Date(1328485501181+0000)\/","errors":{"code":0,"description":"Successful"},"requestid":"d8bf928c-e9f1-45af-8fde-5b68ea76f1cc","responseid":"79f55e46-8b42-4180-a57b-5d5c79c570be"}

The sessiontoken key is the significant item of information from this. A session token is required for every API operation.

Data Node

The key information from every response is contained within the data node. This contains the results of any API Call you might issue.

Error Node

The error code node details the specific error with the Call and is included with every response. An error code of zero (0) means the operation was successful. Negative error codes mean that the Call failed (see Appendix B for a list of current codes). Positive error codes mean that the Call succeeded but there were potential warnings or other issues that should be assessed.

Writing API Calls

As explained earlier, it is possible to author your own API calls in JavaScript. The following is template you can use for writing your own Calls:

```
function [Call Name]([Parameters]) {

    if ([error condition]) {
        this.SetResponseError([error code]);
        return;
    }

    var data = { '[data key]': [value], 'version': success };

    this.SetResponseData(data);
}
```

Example:
This is the newassetversion call code:

```
function newassetversion(assetid) {
    var id = FocusOPEN.Shared.NumericUtils.GetIntValue(ToInt32(assetid), 0);

    var asset = FocusOPEN.Data.Asset.Get(id);
    var version = 0;

    if (asset.IsNull) {
        this.SetResponseError(-10004); //asset does not exist
        return;
    }

    FocusOPEN.Business.AssetVersionManager.Instance.SaveToNewVersion(asset,
this.getUser());
    version = asset.VersionNumber;

    var data = { 'assetid': assetid, 'version': success };

    this.SetResponseData(data);
}
```

Script Storage & Access
Scripts are stored in the folder: Scripts\REST\Assets

The format for accessing the Call is

[URL]/scripts.REST.[folder].[call].ashx?userAPIToken=[User API Token]&[parameter name]=[value]

Example, this would call the process pending imports Call which will automatically ingest all assets for users in brand 1 that had been placed into the /data/imports/[brand]/[user]/manual folder

http://localhost/focusopen/scripts.REST.assets. processpendingimports.ashx ?SessionAPIToken=5af78503-e826-4267-a4c3-416619499f1a&brandid=1

Nullable Integers
If you make Calls that reference integers like Asset IDs, you will find you cannot use them directly without first converting the type to nullable integers (these are integers that can be used with databases). There is no native JavaScript method for doing this, but one has been provided in FocusOPEN which you can use via this template:

```
FocusOPEN.Shared.NumericUtils.GetIntValue(ToInt32([Integer]), 0);
```

Pre-Built Example API Calls

At the time of writing there are the following pre-built API calls.

Assets

newassetversion	Ceates a new version of a given asset
processpendingimports	Processes any assets added to the user's manual imports folder
resubmitprocessing	Re-submits assets for processing

Homepage

gethomepagewidgets	Retrieves custom home page widgets that may have been populated by admin users

Lightboxes

getlightboxdetails	Retrieves details of all assets in a given lightbox

Orders

getorderdetails	Retrieves details of all assets in a given order

Orders

getorderdetails	Retrieves details of all assets in a given order

Triggers

getorderdetails	Called when an asset is published. By default notifies all super admin users that an asset has been published.

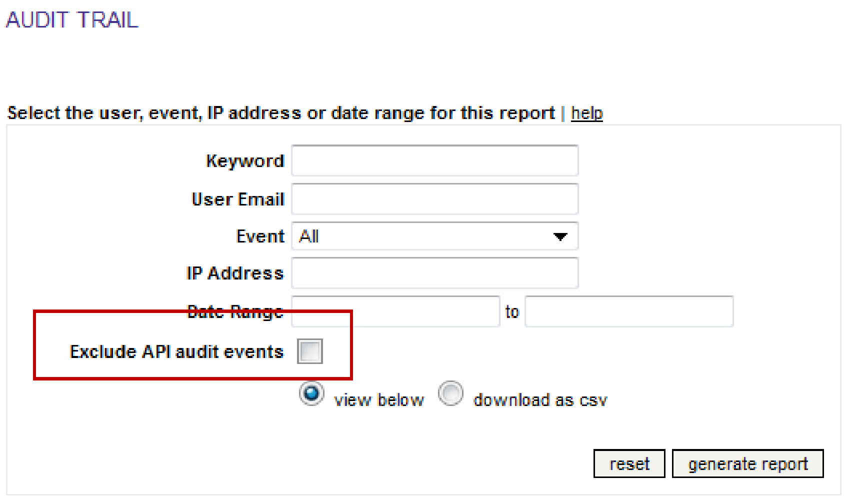
Users

login	Login function discussed earlier in this document.
Logout	Logs the user out of the session (requires a new Session Token to be generated)

Forthcoming releases of FocusOPEN will add to the built-in API calls with additional functionality to enable more comprehensive control of FocusOPEN by REST clients (in addition to any functionality you may author yourself).

Audit Trail & Logging

All API Calls are added to the application Audit Trail as with any other FocusOPEN operation (this is the audit trail that can be accessed by users from Admin/Reports/AuditTrail.aspx). The notes field is prefixed with API Call and contain details of the URL called. The audit trail default is to filter out API calls but they can be enabled again by unchecking "Exclude API audit events":


AUDIT TRAIL

Select the user, event, IP address or date range for this report | help

Keyword	
User Email	
Event	All
IP Address	
Date Range	to
Exclude API audit events	☐

[✓ view below] download as csv

All Calls and Responses are also serialised into XML and stored in the file system in a separate API log folder (outside the webroot).

Every API Call and Response should also be written to the general application logs so it can be reviewed along with other system faults.

The API is used in this way to generate PDF contact sheets in the lightbox and order areas (using the wkhtmltopdf application to render the actual PDF).

Appendix A - REST API Settings

It is possible to set the scope of the API with some settings:

APIRestrictAccounts	If empty, any account can access the API. If a comma separated list of email addresses is specified, only those accounts with logins that are in this list are permitted to use the API
APIObeyIPAddressRestrictions	Set to True by default. This means that any API calls must originate from either the application or one of the approved source IP addresses defined in the Manage IP Addresses area. If the IP address restrictions are off, this setting is ignored. If they are enabled, setting this to False would bypass the IP address restrictions and allows access from other IP addresses.

All of these settings override the standard security using API and Session Tokens and are in place to enable API access to be more strictly controlled if required.

Appendix B - API Error Codes

The API errors are all defined in an XML file called APIErrors.xml

If you add to this file, you are strongly recommended to add any custom errors in the range -20000 and beyond to reduce the potential for conflict with any new errors that are added in updated editions.

```xml
<APIErrors>
    <!-- Generic -->
    <Error id="-1" description="Request method must be GET for this call" />
    <Error id="-2" description="Request method must be POST for this call" />
    <Error id="-3" description="Response format not permitted (must be XML or JSON)" />
    <Error id="-4" description="Session token is invalid" />
    <Error id="-5" description="User does not have permissions to issue that Call" />
    <Error id="-6" description="User does not have permissions to access that asset" />
    <Error id="-7" description="Call failed for an unknown reason" />

    <Error id="-10" description="Call failed for an unknown reason" />

    <!-- REST API Specific -->
    <Error id="-10000" description="UserAPIToken not recognised or API access not
permitted" />
    <Error id="-10001" description="User account has been suspended" />
    <Error id="-10002" description="Account has expired" />
    <Error id="-10003" description="IP address not permitted" />
    <Error id="-10004" description="Requested asset does not exist.  Asset may have
been deleted or is out of range for this instance, check the ID" />
    <Error id="-10005" description="Asset ID not supplied and is required for this
operation" />
    <Error id="-10006" description="Asset file cannot be found" />
    <Error id="-10007" description="Order ID not specified" />
    <Error id="-10008" description="Lightbox ID not specified" />

    <Error id="-10010" description="Script file not found" />
    <Error id="-10011" description="Script folder not found" />
    <Error id="-10012" description="Invalid URI format" />

    <Error id="-10013" description="Brand ID not specified or invalid" />
    <Error id="-10014" description="No users were found or unauthorised access
attempted" />

    <!-- User defined -->

</APIErrors>
```