Fabasoft

White Paper

Fabasoft Folio Friendly URLs

Fabasoft Folio 2026 Update Rollup 2

Copyright © Fabasoft R&D GmbH, Linz, Austria, 2026. All rights reserved. All hardware and software names used are registered trade names and/or registered trademarks of the respective manufacturers.

No rights to our software or our professional services, or results of our professional services, or other protected rights can be based on the handing over and presentation of these documents.

Contents

1 Introduction 3

2 Software Requirements 3

3 Predefined Friendly URLs 3

3.1 download 3

3.2 downloadzip 4

3.3 read 4

3.4 readraw 5

3.5 image 5

3.6 convert 6

3.7 thumbnail 7

3.8 mbzsearch 8

4 Configuration 9

4.1 Friendly URLs Based on Web Service Definitions 9

4.1.1 Web Service Definitions 9

4.1.2 Friendly URL 9

4.2 Friendly URLs Configured in FSCOWS@1.1001:WebServiceConfiguration 10

5 Request Parameters 14

5.1 Request Parameters in a HTTP GET Request 14

5.2 Request Parameters in a HTTP POST Request 15

5.3 Reserved Request Parameters 15

6 Transaction Variables 15

7 Outgoing Content Type Determination 17

8 Fabasoft app.ducx Example 17

9 Fabasoft app.ducx Code Snippets 18

10 User Language 19

1 Introduction

A friendly URL allows easy access to methods of web resources. It consists of the standard Fabasoft Folio Web Service URL, followed by a defined string. Required parameters follow divided by a slash.

Example: `http://fscwebserver/fsc/rename/COO.1.506.2.5/newname`

This example changes the object name of the object with the given object address to "newname". The implementation of this sample is described in the following chapters.

A more sophisticated example would be a HTTP client, which sends an RTF document to the friendly URL `convert` to convert it to a defined target format.

Example: `http://fscwebserver/fsc/convert/rtf/pdf`

In this example `rtf` is used as the source format and `pdf` as the target format of the conversion. The result of the conversion is transmitted back in the HTTP body of the HTTP response.

2 Software Requirements

System environment: All information contained in this document implicitly assumes a Microsoft Windows environment or a Linux environment.

Supported platforms: For detailed information on supported operating systems and software see the software product information on the Fabasoft distribution media.

3 Predefined Friendly URLs

Fabasoft Folio provides some predefined friendly URLs that can be used right away.

3.1 download

Allows downloading a content. The `Content-Disposition` header is set to `attachment` to enforce the presentation the content as downloadable file.

Friendly URL

```
http://<web server>/<vdir>/download/<objectid>/
  <contentpath (optional)>/<downloadname (optional)>
```

Parameters:

- `objectid`
  The COO address of the object, which contains the content that should be downloaded.
- `contentpath`
  The property path defining the content. The default value is: `content(0)contcontent(0)`
- `downloadname`
  The name that should be used as file name. The default value is the object name.

Example

```
http://server.mycomp.com/fsc/download/COO.1.3285.3.9
http://server.mycomp.com/fsc/download/COO.1.3285.3.9/contentfinalform%280%29contcontent%280%29/test.pdf
```

3.2 downloadzip

Allows downloading several contents as ZIP file.

Friendly URL

```
http://<web server>/<vdir>/downloadzip/<objects>
```

Parameters:

- `objects`
  A list of COO addresses separated by a semicolon. If only one object is provided, the object itself will be downloaded and no ZIP file is generated.

Example

```
http://server.mycomp.com/fsc/downloadzip/COO.1.3285.3.9
http://server.mycomp.com/fsc/downloadzip/COO.1.3285.3.9;COO.1.3285.3.12
```

3.3 read

Allows retrieving a content. Document properties (if present) are evaluated.

Friendly URL

```
http://<web server>/<vdir>/read/<objectid>/
  <contentpath (optional)>/<maxageseconds (optional)>
```

Parameters:

- `objectid`
  The COO address of the object, which contains the content that should be read.
- `contentpath`
  The property path defining the content. The default value is: `content(0)contcontent(0)`
- `maxageseconds`
  Sets the HTTP response header field `Cache-Control` to the defined value `max-age=<maxageseconds>`.

Example

```
http://server.mycomp.com/fsc/read/COO.1.3285.3.9
```

```
http://server.mycomp.com/fsc/read/COO.1.3285.3.9/content%280%29contcontent%280%29/
50000
```

3.4 readraw

Allows retrieving a content. Document properties are not evaluated.

Friendly URL

```
http://<web server>/<vdir>/readraw/<objectid>/
  <contentpath (optional)>/<maxageseconds (optional)>
```

Parameters:

- `objectid`
  The COO address of the object, which contains the content that should be downloaded.
- `contentpath`
  The property path defining the content. The default value is: `content(0)contcontent(0)`
- `maxageseconds`
  Sets the HTTP response header field `Cache-Control` to the defined value `max-age=<maxageseconds>`.

Example

```
http://server.mycomp.com/fsc/readraw/COO.1.3285.3.9

http://server.mycomp.com/fsc/readraw/COO.1.3285.3.9/content%280%29contcontent%280%
29/50000
```

3.5 image

Allows retrieving an image of an object. Available image types are configured in the *Conversion Configuration* in the *Image Type Configuration* field.

Friendly URL

```
http://<web server>/<vdir>/image/<objaddr>/<name>/<page (optional)>
```

Parameters:

- `objaddr`
  The COO address of the object, which contains the content that should be retrieved as image.
- `name`
  Defines the kind of image. Possible values are defined in the *Conversion Configuration*, on the "Conversion Configuration" tab, in the *Image Type Configuration* field (e.g. `tn` or `pv`).
- `page`
  In the *Image Type Configuration* it can be defined that more than one page is generated as image. The `page` parameter can be used to get the desired page (e.g. `2`).

Example

```
http://server.mycomp.com/fsc/image/COO.1.3285.3.9
http://server.mycomp.com/fsc/image/COO.1.3285.3.9/pv/1
```

3.6 convert

Allows converting a content into another format (HTTP POST request).

Friendly URL

```
http://<web server>/<vdir>/convert/<sourcefmt>/<destfmt>
```

Parameters:

- `sourcecont` (content input parameter)
  The source content with content type `application/octet-stream`.
- `destcont` (content output parameter)
  The target content. The content type is defined by `destfmt`.
- `sourcefmt`
  Typically a file extension (e.g. `doc`).
- `destfmt`
  Typically a file extension (e.g. `pdf`).

The parameters `sourcefmt` and `destfmt` are evaluated based on the *Conversion Configuration*.

Example (Java)

```
// Fabasoft Folio Friendly URL scheme is used for the end point URL
URL endPoint = new URL(FolioWebSvcSample.WEBSERVER + "/convert/" +
  sourceFormat + "/" + destFormat);
HttpURLConnection connection = (HttpURLConnection) endPoint.openConnection();
// set authentication if required
if (FolioWebSvcSample.AUTHENTICATION == "BASIC") {
  Authenticator.setDefault(new Authenticator() {
    protected PasswordAuthentication getPasswordAuthentication() {
      return new PasswordAuthentication(FolioWebSvcSample.USERNAME,
        FolioWebSvcSample.PASSWORD.toCharArray());
    }
  });
}
// set request properties
connection.setRequestMethod("POST");
connection.setRequestProperty("Content-Type", "application/octet-stream");
connection.setDoInput(true);
connection.setDoOutput(true);
connection.setUseCaches(false);
// connect to web server
connection.connect();
byte[] buf = new byte[2048];
int received = 0;
// read from the source file and write to the connection output stream (upload)
FileInputStream sourceFileStream = new FileInputStream(sourceFile);
OutputStream uploadStream = connection.getOutputStream();
```

```
while ((received = sourceFileStream.read(buf, 0, buf.length)) > 0) {
  uploadStream.write(buf, 0, received);
}
uploadStream.flush();
uploadStream.close();
sourceFileStream.close();
// read from the connection input stream and write to destination file (download)
FileOutputStream destFileStream = new FileOutputStream(destFile);
InputStream downloadStream = connection.getInputStream();
while ((received = downloadStream.read(buf, 0, buf.length)) > 0) {
  destFileStream.write(buf, 0, received);
}
destFileStream.flush();
destFileStream.close();
```

3.7 thumbnail

Allows generating a thumbnail of a content (HTTP POST request).

Friendly URL

```
http://<web server>/<vdir>/thumbnail/<srctype>/<bbwidth>/<bbheight>/
  <pageindex>/<thumbtype>
```

Parameters:

- `srccont` (content input parameter)
  The source content with content type `application/octet-stream`.
- `thumbnail` (content output parameter)
  The generated thumbnail. The content type is defined by `thumbtype`.
- `srctype`
  Typically a file extension (e.g. `doc`).
- `bbwidth`
  Width of the thumbnail.
- `bbheight`
  Height of the thumbnail.
- `pageindex`
  The page inside a document.
- `thumbtype`
  Typically a file extension (e.g. `jpg`).

Example (C#)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.IO;

namespace HttpRequest
{
  class Program
```

```
  {
    static void Main(string[] args)
    {
      string target = "http://localhost/fsc/thumbnail/cr2/250/250/1/jpg";
      string requestfile = @"C:\friendlyurltester\RAW_CANON_1DSM3.CR2";
      string httpmethod = "POST";
      Uri targetUri = new Uri(target);
      HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(targetUri);
      request.Method = httpmethod;
      // BASIC authentication
      string user = "Username";
      string pass = "Password";
      string authStr = user + ":" + pass;
      request.Headers.Add("Authorization", "Basic " +
        Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(authStr)));
      // read http body from file
      byte[] buf = null;
      try {
        FileStream fs = new FileStream(requestfile, FileMode.Open);
        buf = new byte[fs.Length];
        fs.Read(buf, 0, (int)fs.Length);
      }
      catch (Exception) {
      }
      if (buf == null || buf.Length == 0) {
        request.ContentLength = 0;
      }
      else {
        request.ContentLength = buf.Length;
      }
      // send request
      if (buf != null && buf.Length > 0) {
        Stream instream = request.GetRequestStream();
        instream.Write(buf, 0, buf.Length);
      }
      // download response
      WebResponse response = request.GetResponse();
      Stream outstream = response.GetResponseStream();
      FileStream outfs = new FileStream(@"c:\friendlyurltester\thumbnail.jpg",
        FileMode.CreateNew);
      byte[] outbuf = new byte[1024];
      int bytesread = 0;
      while ((bytesread = outstream.Read(outbuf, 0, outbuf.Length)) > 0) {
        outfs.Write(outbuf, 0, bytesread);
      }
      outfs.Close();
    }
  }
}
```

3.8 mbzsearch

Redirects to the Mindbreeze search portal.

The value of the input text field of the Mindbreeze search portal is prefilled with the value from the URL parameter `userquery`. If the URL parameter `startsearch` is set to "1", the Mindbreeze search is started after the redirect. If the Mindbreeze search portal is not available, the user is redirected to the home location ("base URL").

Friendly URL

```
https://<web server>/<vdir>/mbzsearch?userquery=<searchquery
(optional)>&startsearch=<startsearch (optional)>
```

Parameters:

- `userquery(optional)`
  The text of the search query, which is prefilled in the input text field of the Mindbreeze search portal.
- `startsearch(optional)`
  Defines whether the Mindbreeze search should be started after the redirect.
  Allowed values are: "1" or "0".

4 Configuration

Friendly URLs can either be based on web service definitions or explicitly configured in `FSCOWS@1.1001:WebServiceConfiguration`.

4.1 Friendly URLs Based on Web Service Definitions

This chapter describes friendly URLs based on web service definitions.

4.1.1 Web Service Definitions

A web service definition defines a set of actions (or SOAP actions), which are bundled in a single accessible endpoint represented by a WSDL. Additionally RESTful services/operations are exposed, too. The list of operations exposed by the web service is defined in `FSCOWS@1.1001:webserviceactions`.

The format of a friendly URL based on a web service definition looks like:

`http(s)://localhost/fsc/MYCOMP_620_1200_WebService.MyOperation`

- `MYCOMP@620.1200:WebService`
  reference of a web service definition
- `MyOperation`
  a specific web service operation

Note: References in URLs must be written in underscore notation. If the software component reference contains underscores, these are duplicated. The characters "@", "." and ":" are substituted by underscores. For example, the reference `MY_COMP@620.1200:Web_Service` results in `MY__COMP_620_1200_Web_Service`.

4.1.2 Friendly URL

In case that only the operations defined in a `FSCOWS@1.1001:WebServiceDefinition` should be exposed for friendly URL access no explicit configuration is necessary, if the following applies:

- The name of the friendly URL is `<full reference of web service definition in underscore notation>.<web service operation or short reference of action>`.

- The parameters of the actions defined in `FSCOWS@1.1001:webserviceactions` only contain types that are usable in friendly URLs.
- The parameters of the actions defined in `FSCOWS@1.1001:webserviceactions` have the same sequence as the parameters of the friendly URLs.
- The actions defined in `FSCOWS@1.1001:webserviceactions` have zero or one input content.
- The actions defined in `FSCOWS@1.1001:webserviceactions` have zero or one output content.
- The service supports only `HTTP_GET` and `HTTP_POST` (`HTTP_POST` only if an action defined in `FSCOWS@1.1001:webserviceactions` has an input content).

4.2 Friendly URLs Configured in FSCOWS@1.1001:WebServiceConfiguration

Friendly URLs can be explicitly configured in the `FSCOWS@1.1001:WebServiceConfiguration`.

Following parameters can be defined:

- *Type*
  The type determines how parameters of the REST call are mapped to the implementation.
- *URL Marker*
  The primary part of the friendly URL.
- *URL Arguments (Necessary Type: OpenAPI)*
  Contains static URL parts, named URL parameters and named query parameters.
  For example: `/{arg1}?{urlarg1}`
  Match URL: `http://localhost/fsc/documentapi/123?urlarg1=true`
  Defines the URL parameter `arg1` and the query parameter `urlarg1`.
  For example: `/v2/{arg1}/{arg2}?{urlarg1}{urlarg2}`
  Match URL: `http://localhost/fsc/documentapi/v2/123/456?urlarg1=true&urlarg2=FK7`
  Defines the URL parameters `arg1` and `arg2` and the query parameters `urlarg1` and `urlarg2`.
  - **Match URLs**
    Request URLs are matched using
    1. `FSCOWS@1.1001:friendlyurlurl`
    2. `FSCOWS@1.1001:friendlyurlallowedmethods`
    3. `FSCOWS@1.1001:friendlyurlargs` static URL parts

    If multiple static prefixes match, the entry with the most matching static URL parts is used.
    `FSCOWS@1.1001:friendlyurlallowedmethods`
    The request method has to match the specified methods.
    `FSCOWS@1.1001:friendlyurlaction`
    The arguments for the action are matched by name.
    `FSCOWS@1.1001:friendlyurloutcont`
    If set, the value of this parameter is used as response content.
  - **Input Parameters**
    Input parameters are resolved by name from:
    1. Named URL parameters
    2. Named query parameters
    3. Request content

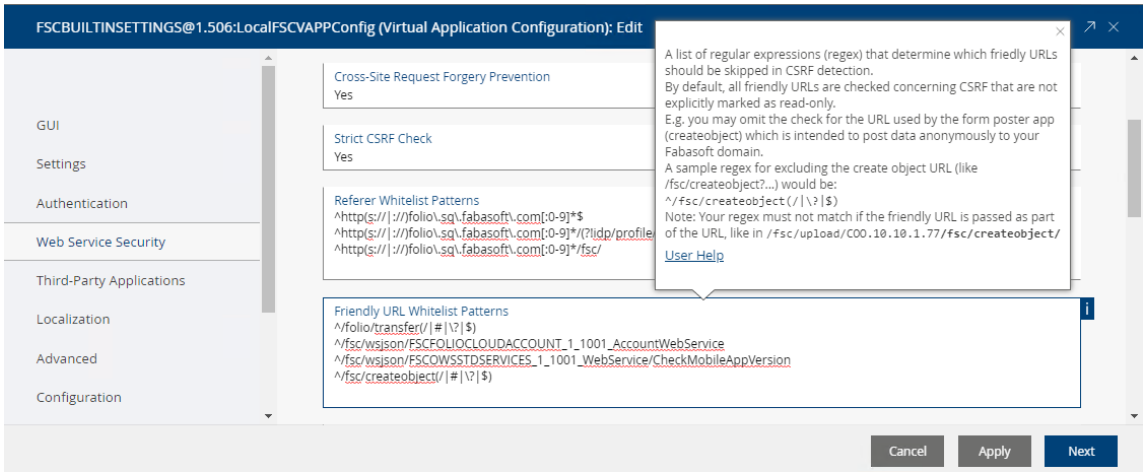
JSON (Content-Type: "application/json; charset=utf-8")
The document is parsed and named parameters are resolved by name based on parameter types.
Basic JSON Types: The value is matched according to `FSCEXPEXT@1.1001:JSON2Value`.
`COOSYSTEM@1.1:DATETIME`: The value is matched according to `xs:DateTime`. e.g. "2001-10-26T21:32:52".
`FSCOWS@1.1001:NamedPart`: The string value has to match the filename of an attached document.
Other aggregate type: The value is expected to be structured. Every structure member is resolved as described.
Values must match the corresponding type; enumeration values are matched by reference.

- **Output Parameters**
  Output parameters are added by name and to a result JSON document "Content-Type" "application/json; charset=utf-8".
  The transformation of values is implemented according to the resolving of input parameters.

- *DAV Action*
  The action to be executed.
- *Content Input Parameter*
  Contains the HTTP body of the HTTP request. If present, the content is passed to the DAV action for processing. The parameter is passed as a type `COOSYSTEM@1.1:CONTENT`.
- *Content Output Parameter*
  If the DAV action returns one or more contents as result, this parameter is used to define which content is used as HTTP body of the HTTP response. The parameter must be of type `COOSYSTEM@1.1:STRING`, `COOSYSTEM@1.1:CONTENT`, `COOSYSTEM@1.1:CONTENTLIST` or `COOSYSTEM@1.1:Content`. In the latter case, the value in the contained attribute `COOSYSTEM@1.1:contextension` is used to determine the MIME type of the outgoing HTTP response.
- *Parameter Mapping (Necessary Type: Friendly URL)*
  Maps friendly URL parameters to the parameter of the DAV action. More specific, the name of the action parameter is mapped to the position of the value in the friendly URL. The object on which the DAV action is called can also be specified using the keyword "this" as action parameter. In this case, the specific URL parameter has to be a valid object address.
  Example: `http://localhost/fsc/friendlyurl/param1/COO.1.1.1.1/param2`
  If the second URL parameter has been configured with the action parameter "this", the object with the object address `COO.1.1.1.1` is used as the object on which the DAV action is called upon.
- *Allowed HTTP Methods*
  Each friendly URL call is initially restricted to the HTTP methods "GET", "HEAD" and "OPTIONS". To extend this restriction, additional HTTP methods can be configured within this attribute.
  Example: For the "convert" friendly URL to function properly, the HTTP method "POST" must be configured.

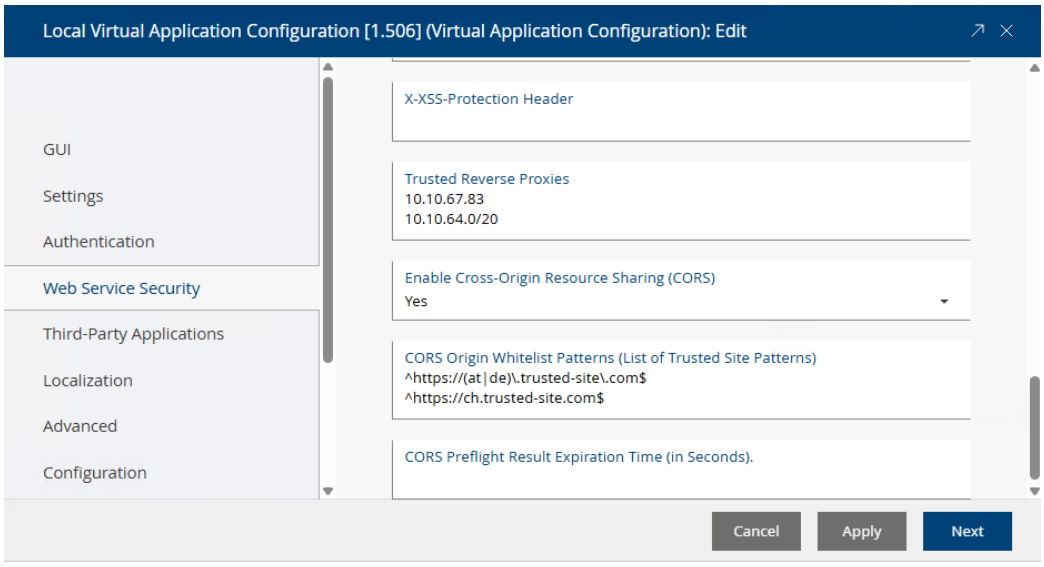
Should a non-configured HTTP method be used in a request, the request is answered with HTTP 405 "Not Allowed".

- *Action Will Not Change State*
  This setting only applies if "Cross Site Request Forgery" (CSRF) prevention is configured in the virtual application configuration (see `FSCVAPP@1.1001:csrf` and `FSCVAPP@1.1001:csrffurlwhitelist` of the `FSCVAPP@1.1001:DefaultConfiguration` for more details).
  If the action called by the friendly URL does not make changes, you may set this field to `true`. In that case the CSRF check is turned off for this kind of URLs. CSRF attacks specifically target state-changing requests and not theft of data, since the attacker has no way to see the response to the forged request. Therefore, if the friendly URL action (`FSCOWS@1.1001:friendlyurlaction`) or actions called by the friendly URL action will change any state this property **must not** be set to true.
  Sample configuration for CSRF with a friendly URL whitelist:



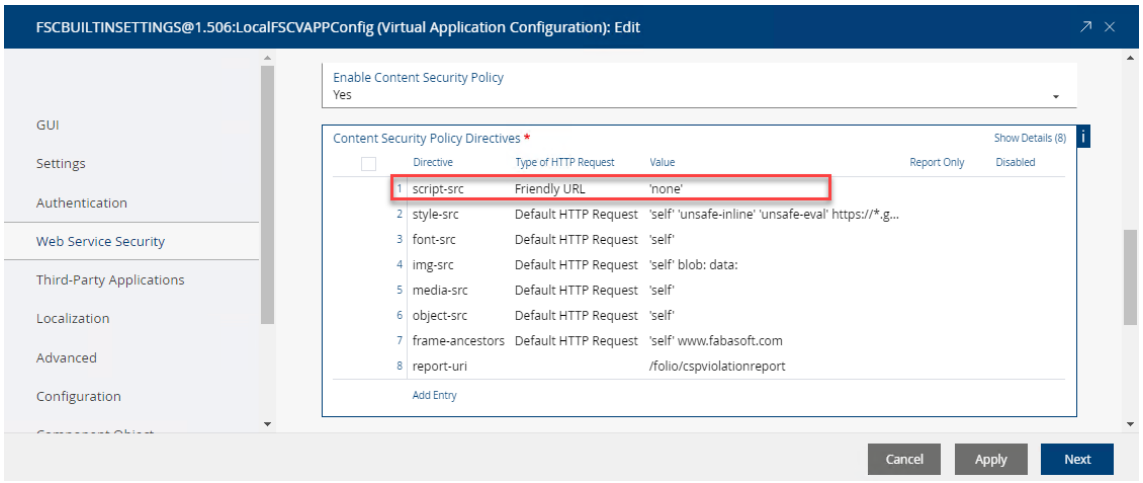
- *Indicates whether Cross-Origin Resource Sharing (CORS) is enabled*
  This setting defines whether Cross-Origin Resource Sharing (CORS) is allowed for the friendly URL. This setting only applies if Cross-Origin Resource Sharing (CORS) is enabled in the virtual application configuration. For the user agent (cross-origin web client) to use CORS, the origin (scheme, hostname, and port) must be trusted. This can be achieved by defining a regular expression (regex) and adding it to the list of trusted sites using the `FSCVAPP@1.1001:corsoriginwhitelist` property. For more information, see `FSCVAPP@1.1001:corsenabled` and `FSCVAPP@1.1001:corsoriginwhitelist` in FSCVAPP@1.1001:DefaultConfiguration.
  Note that CORS is not currently supported for the OpenAPI friendly URL type.

Sample configuration for CORS whitelist:



- **Multiple trusted sites:** The origins `https://at.trusted-site.com` and `https://de.trusted-site.com` are allowed to use CORS because they are matching the regular expression pattern: `^https://(at|de)\.trusted-site\.com$` defined in the `FSCVAPP@1.1001: corsoriginwhitelist` property.
- **Single trusted site:** The website `https://ch.trusted-site.com` is allowed to use CORS because it is matching the regular expression pattern: `^https://ch.trusted-site.com$` defined in the `FSCVAPP@1.1001: corsoriginwhitelist` property.
  Note: To define a single trusted site, it is important that the regular expression **starting character ^** and the **ending character $** are correctly set. Otherwise, origins not matching exactly but matching the specified part might be also considered as trusted.
- *HTTP Response Requires Script Execution*
  This setting only applies if Content Security Policy (CSP) settings are configured in the Virtual Application Configuration (see property `FSCVAPP@1.1001:cspdirectives` of `FSCVAPP@1.1001:DefaultConfiguration`).
  Special CSP directives may be applied for friendly URL requests (e.g. you may configure that the execution of scripts should be disabled when executing requests for friendly URLs). You may set this option to `true` if the CSP directives of ordinary HTTP requests should be used for requests for this friendly URL, too. For a detailed description to Content Security Policy refer to: http://www.w3.org/TR/CSP2/
  Sample of an appropriate CSP configuration that will not allow the execution of scripts

for friendly URLs:



- *Software Component*
  The software component that has added the entry.
- *Friendly URL CSP Request Type* (`FSCVAPP@1.1001:friendlyurlcsprequesttype`)
  By using the content security policy (CSP) settings (see property `FSCVAPP@1.1001:cspdirectives` of `FSCVAPP@1.1001:DefaultConfiguration`) CSP directives can be applied for friendly URL requests.
  For example, you can set this option to "Default HTTP Request" if you want the CSP directives of "normal HTTP requests" to be used for requests to this friendly URL.
  This setting only applies if content security policy (CSP) settings are configured in the *Virtual Application Configuration* (see property `FSCVAPP@1.1001:cspdirectives` of `FSCVAPP@1.1001:DefaultConfiguration`).
  Available types:
- Default HTTP Request
  Applies CSP directives of "normal HTTP requests".
- Friendly URL
  Applies CSP directives of friendly URLs.
- Office JS Integration
  Applies CSP directives of the Office JS integration.

5 Request Parameters

This chapter describes the request parameters.

5.1 Request Parameters in a HTTP GET Request

In addition to a slash being the divider between the URL marker and the parameters, a question mark can be used. In this case the parameter mapping in the *Friendly URL Configuration* (`FSCOWS@1.1001:WebServiceConfiguration`) will be ignored.

If a question mark is used as divider, the following string is interpreted as a URL query string. The implementation then tries to parse the parameters assuming each parameter (or key-value pair) is separated by a `"&"` character and spitted by "=". Both key and value are URL-Decoded

before they are inserted to the Dictionary. The parsed values are then passed to the DAV action by the transaction variable `TV_FRIENDLYURL_URL_PARAMS` as a `COOSYSTEM@1.1:DICTIONARY`.

Example:

URL: `http://localhost/fsc/friendlyurl?param1=123&param2=value2&param3=COO.1.1.1.1`

Value of dictionary `TV_FRIENDLYURL_URL_PARAMS`:
key: `param1`, value: `123`
key: `param2`, value: `value2`
key: `param3`, value: `COO.1.1.1.1`

Values in this dictionary are of type `COOSYSTEM@1.1:STRING`

5.2 Request Parameters in a HTTP POST Request

Request parameters can be sent in a HTTP POST request in the HTTP body. This is usually used when a HTML POST form is sent to the web server. The values are provided as key/value pairs similar to a HTTP GET request. The HTTP body can be encoded in two ways:

URL encoded form data

The parameters are encoded similar to those of the HTTP GET request.

Multipart form data

The parameters are encoded as multipart type. This is usually used when a file is to be uploaded to the web server. The file is stored at the web server and deleted when the HTTP request is finished. The absolute path to the file is provided as value in the resulting dictionary.

In both cases it is assumed that non-ASCII characters are properly encoded as UTF-8. By specifying the attribute `accept-encoding="UTF-8"` in the HTML form it is ensured that the characters are encoded properly.

5.3 Reserved Request Parameters

To utilize certain features of the vApp Engine, a certain URL marker is used to mark certain URL parts as only relevant for the vApp Engine itself.

Everything in the URL following the defined parameter `"&vappquerystring=true"` is considered as an argument for the vApp Engine and therefore not interpreted by the Friendly URL parser.

Example:

```
http://localhost/fsc/friendlyurl?param1=v1&param2=v2&vappquerystring=true&lx=en&..
http://localhost/fsc/friendlyurl/param1/param2&vappquerystring=true&lx=en&...
```

Only the value of "param1" and "param2" is passed to the friendly URL implementation.

6 Transaction Variables

When developing a DAV action, several transaction variables of software component `FSCASP@1.1001` can be used:

- `TV_FRIENDLYURL_FILEEXTENSION`
  Type: `COOSYSTEM@1.1:STRING`
  Overrides the outgoing file extension and is also used to determine the outgoing MIME type. See chapter 6 for further details.
- `TV_FRIENDLYURL_ADDITIONAL_HEADERS`
  Type: `COOSYSTEM@1.1:STRINGLIST`
  Additional HTTP headers can be added to the outgoing HTTP request by adding them to this transaction variable. The HTTP header has to be formatted correctly.
  Example: `X-MyCustomHeader: headervalue`
- `TV_FRIENDLYURL_RESPONSE_AS_DOWNLOAD`
  Type: `COOSYSTEM@1.1:STRING`
  This causes the outgoing Content-Disposition HTTP header to be set to `DOWNLOAD`. The value of the transaction variable is the filename used within the HTTP header. A `DOWNLOAD` setting usually causes the browser to display a download dialog.
- `TV_FRIENDLYURL_RESPONSE_AS_INLINE`
  Type: `COOSYSTEM@1.1:STRING`
  This causes the outgoing Content-Disposition HTTP header to be set to `INLINE`. The value of the transaction variable is the filename used within the HTTP header. An `INLINE` setting causes the browser to try to render the received content accordingly.
  Example: A PDF document is rendered inline using the Adobe browser plug-in.
- `TV_FRIENDLYURL_RAW_CONTENT`
  Type: `COOSYSTEM@1.1:BOOLEAN`
  The outgoing content will not be patched with DocProperties, if this transaction variable is set to `TRUE`.
- `TV_FRIENDLYURL_INCOMING_HEADERS`
  Type: `COOSYSTEM@1.1:DICTIONARY`
  The incoming HTTP headers can be read from this transaction variable as a `DICTIONARY` key value pair.
- `TV_FRIENDLYURL_INCOMING_METHOD`
  Type: `COOSYSTEM@1.1:STRING`
  The incoming HTTP method can be read from this transaction variable.
- `TV_FRIENDLYURL_BASELOCATION`
  Type: `COOSYSTEM@1.1:STRING`
  The location of the friendly URL can be read from this transaction variable.
  Example: `"/fsc/read"` or `"/fsc/convert"`
- `TV_FRIENDLYURL_OUTGOING_STATUSCODE`
  Type: `COOSYSTEM@1.1.INTEGER`
  The DAV action can dictate the HTTP status code which is used for the outgoing HTTP response.
- `TV_FRIENDLYURL_INCOMING_REQUESTURL`
  Type: `COOSYSTEM@1.1:STRING`
  The request string can be read from this transaction variable.
- `TV_FRIENDLYURL_URL_PARAMS`
  Type: `COOSYSTEM@1.1:DICTIONARY`

If URL parameters have been identified, the parsed key-value pairs are provided within this dictionary as `COOSYSTEM@1.1:STRING`.

- `TV_FRIENDLYURL_URL_PARTS`
  Type: `COOSYSTEM@1.1:DICTIONARY`
  If URL parts have been identified, the parsed values are provided within this dictionary as `COOSYSTEM@1.1:STRING`. The parsed values can be accessed by the sequence number.

7 Outgoing Content Type Determination

In order to determine the value for the outgoing content type HTTP header, a certain priority exists:

1. Custom `Content Type` header from the transaction variable `TV_FRIENDLYURL_ADDITIONAL_HEADERS`.
2. File extension from the transaction variable `TV_FRIENDLYURL_FILEEXTENSION`.
3. Value of `COOSYSTEM@1.1:content.COOSYSTEM@1.1:contencoding`.
4. Value of `COOSYSTEM@1.1:content.COOSYSTEM@1.1:contextension`.
5. File extension from outgoing file.

8 Fabasoft app.ducx Example

The following example shows how to define an action in Fabasoft app.ducx and how to add a friendly URL to the configuration.

Example

app.ducx Use Case Language

```
usecases SAMPLERENAME@200.200
{
  import COOSYSTEM@1.1;
  RenameObject(string objid, string newname) {
    variant Object {
      impl = expression {
        Object(objid).objname = newname;
      }
    }
  }
}
```

app.ducx Object Model Language

```
// The following example shows how to add the friendly URL to the configuration.
objmodel SAMPLERENAME@200.200
{
  import FSCOWS@1.1001;
  extend instance FSCOWS@1.1001:WebServiceConfiguration {
    friendlyurlconfig<friendlyurlurl,
                      friendlyurlaction,
                      friendlyurlincont,
                      friendlyurloutcont,
                      friendlyurlparams<friendlyurlparamnr,friendlyurlparamname>,
                      friendlyurlswc
```

```
                        > =
    {
      {
        "rename",
        RenameObject,
        "",
        "",
        { {1, "objid"}, {2, "newname"} },
        SAMPLERENAME@200.200
      }
    }
  }
}
```

9 Fabasoft app.ducx Code Snippets

Calling path: http://webserver/fsc/statuscode/200

Example

app.ducx Object Model Language

```
extend instance FSCOWS@1.1001:WebServiceConfiguration {
  friendlyurlconfig<friendlyurlurl,
                    friendlyurlaction,
                    friendlyurlincont,
                    friendlyurloutcont,
                    friendlyurlparams<friendlyurlparamnr,friendlyurlparamname>,
                    friendlyurlallowedmethods
                    > =
  {
    {
      "statuscode",
      FriendlyStatusCode,
      "",
      "response",
      { {1, "statuscode"} },
      HTTPMETHOD_OPTIONS
    }
  }
}
```

app.ducx Use Case Language

```
FriendlyStatusCode(integer statuscode, out string response) {
  variant Object {
    impl = expression {
      coort.Trace("FriendlyStatusCode: outgoing status code", statuscode);
      response = "<text>This is a string Text</text>";
      // The Default Content Type Response in string mode is
      // "text/html; charset=utf-8"
      // Set Content-Type to: "text/xml"
      #TV.FSCASP@1.1001:TV_FRIENDLYURL_FILEEXTENSION = "xml";
      #TV.FSCASP@1.1001:TV_FRIENDLYURL_OUTGOING_STATUSCODE = statuscode;
    }
  }
}
FriendlyStatusCode(integer statuscode, out content response) {
  variant Object {
    impl = expression {
```

```
      response = coort.CreateContent();
      response.SetContent(cootx, 1, 65001, "<text>This is a content</text>");
      // There is no default Content-Type when delivering a content
      // -- you have to choose one
      #TV.FSCASP@1.1001:TV_FRIENDLYURL_FILEEXTENSION = "xml";
    }
  }
}

FriendlyStatusCode(integer statuscode, out Content response) {
  variant Object {
    impl = expression {
      content contresponse = coort.CreateContent();
      contresponse.SetContent(cootx, 1, 65001, "<text>This is a Content</text>");
      response.contcontent = contresponse;
      // There is no default Content-Type
      // The Content-Type is set from the Value of the contextension
      response.contextension = "xml";
    }
  }
}
```

10 User Language

It has to be considered that in case of Friendly URL requests the `Accept-Language` header of the HTTP request will be ignored. If the current user has not configured a user language in the user environment the default language of the Fabasoft Folio domain will be used.