

FICO® Xpress Optimization

Last update 15 Jan, 2020

8.8.2

REFERENCE MANUAL

DMP Module for Mosel

Developers Guide

FICO® **Decisions**

©2017–2020 Fair Isaac Corporation. All rights reserved. This documentation is the property of Fair Isaac Corporation ("FICO"). Receipt or possession of this documentation does not convey rights to disclose, reproduce, make derivative works, use, or allow others to use it except solely for internal evaluation purposes to determine whether to purchase a license to the software described in this documentation, or as otherwise set forth in a written software license agreement between you and FICO (or a FICO affiliate). Use of this documentation and the software described in it must conform strictly to the foregoing permitted uses, and no other use is permitted.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, please report them to us in writing. Neither FICO nor its affiliates warrant that this documentation is error-free, nor are there any other warranties with respect to the documentation except as may be provided in the license agreement. FICO and its affiliates specifically disclaim any warranties, express or implied, including, but not limited to, non-infringement, merchantability and fitness for a particular purpose. Portions of this documentation and the software described in it may contain copyright of various authors and may be licensed under certain third-party licenses identified in the software, documentation, or both.

In no event shall FICO or its affiliates be liable to any person for direct, indirect, special, incidental, or consequential damages, including lost profits, arising out of the use of this documentation or the software described in it, even if FICO or its affiliates have been advised of the possibility of such damage. FICO and its affiliates have no obligation to provide maintenance, support, updates, enhancements, or modifications except as required to licensed users under a license agreement.

FICO is a registered trademark of Fair Isaac Corporation in the United States and may be a registered trademark of Fair Isaac Corporation in other countries. Other product and company names herein may be trademarks of their respective owners.

FICO® Xpress Optimization

Deliverable Version: A

Last Revised: 15 Jan, 2020

Version 8.8.2

Contents

1	Introduction	1
2	Using the DMP module	2
2.1	Accessing DMP environmental data	2
2.2	Making HTTP Requests	2
2.3	Calling a DMP Component	2
2.4	Calling a DMP Webservice	3
2.5	Calling DMP Manager	3
2.6	Use Outside the Cloud	4
2.7	Error handling	4
2.8	Limitations	5
3	Examples	6
3.1	Calling a REST endpoint on an Xpress Executor component	6
3.2	Calling a REST endpoint on DMP Manager	6
4	Types	8
4.1	The <code>dmpresource</code> type	8
5	Procedures and functions	10
	<code>clearerror</code>	11
	<code>dmphttpdel</code>	12
	<code>dmphttpget</code>	13
	<code>dmphttphead</code>	14
	<code>dmphttppost</code>	15
	<code>dmphttpput</code>	16
	<code>dmpinitcomp</code>	17
	<code>dmpinitmanager</code>	18
	<code>dmpiniturl</code>	19
	<code>dmpinitwebservice</code>	20
	<code>getdmpcompid</code>	21
	<code>getdmplifecycleenv</code>	22
	<code>getdmpsolid</code>	23
	<code>isdmp</code>	24
6	Parameters	25
	<code>dmp_verbose</code>	25
	<code>dmp_max_retries</code>	25
	<code>dmp_retry_error_codes</code>	26
	Appendix	27
A	Contacting FICO	27
	Product support	27
	Product education	27

Product documentation 27

Sales and maintenance 28

Related services 28

FICO Community 28

About FICO 28

Index **29**

CHAPTER 1

Introduction

The `dmp` module allows a Mosel model running in a supported Decision Management Platform (DMP) Component (FICO® Xpress Insight or FICO® Xpress Executor) to interact with components and webservices within the same DMP Solution, and restricted parts of DMP Manager pertaining to the solution. Using the `dmpresource` type supplied by the `dmp` module, you will be able to make HTTP requests to these resources that will be automatically authorized with appropriate credentials, for example:

```
declarations
  mycomponent: dmpresource
end-declarations
dmpinitcomp(mycomponent, "Xpress Executor")
if mycomponent.status<>DMP_OK then
  writeln('Component not found')
else
  httpStatusCode := dmhttpget(mycomponent, "/rest/runtime/status", "status.json")
end-if
```

This manual will not cover how to interact with specific DMP webservices; the developer should consult the documentation for the resource they are trying to call for details of how to formulate the HTTP request and interpret the response.

CHAPTER 2

Using the DMP module

2.1 Accessing DMP environmental data

One of the simplest uses of the `dmp` module is to access information about the component instance that's executing your model. You can use `getdmpcompid`, `getdmposolid` and `getdmplifecycleenv` to retrieve the DMP component ID, solution ID and lifecycle stage respectively. You can also use the `isdmp` function to check whether or not the model is being executed by a DMP component. For example:

```
if isdmp then
  writeln('Component ID:', getdmpcompid)
  writeln('Solution ID:', getdmposolid)
  writeln('Lifecycle Stage:', getdmplifecycleenv)
else
  writeln('Not within DMP')
end-if
```

2.2 Making HTTP Requests

You can use the functions `dmphttpget`, `dmphttppost`, `dmphttpput`, `dmphttpdel` and `dmphttphead` to send HTTP requests to a DMP component, webservice or DMP Manager. These functions work much the same as the corresponding functions in `mmhttp`, except instead of passing a destination URL, you pass a destination `dmpresource` and path relative to this.

In event of an unexpected failure, the request can be retried a number of times until it succeeds, the delay between each retry growing progressively larger. When a `dmphttp` request returns an HTTP status code found in `dmp_retry_error_codes`, it will be retried, up to the number of times specified by `dmp_max_retries`—a value of zero disables the feature.

2.3 Calling a DMP Component

The most common use of the `dmp` module will be to send HTTP requests to another component in the same DMP solution. To do this, you first initialize a `dmpresource` value with the details of the target component by calling `dmpinitcomp`, then use the `dmphttp` functions to send HTTP requests in a similar way to using plain `mmhttp`. For example, in an Xpress Insight app, you can send a webservice request to the **status** endpoint of an Xpress Executor component in the same solution as follows:

```
declarations
  mycomp: dmpresource
end-declarations
dmpinitcomp(mycomp, 'Xpress Executor')
if mycomp.status<>DMP_OK then
```

```
writeln('Failed to find component due to error:', mycomp.lasterror)
exit(1)
end-if

httpStatusCode := dmphttpget(mycomp, '/rest/runtime/status', 'result.json')
if httpStatusCode <> 200 then
    writeln('Error returned by component: ', httpStatusCode)
    exit(1)
end-if
```

The path you pass to the `dmphttp` function will be relative to the root of the component instance URL.

When executed from a DMP component, the `dmp` module will take care of obtaining appropriate authorization credentials and including them in the outgoing HTTP requests.

2.4 Calling a DMP Webservice

You can also use of the `dmp` module to send HTTP requests to a `SERVERLESS_REST` type webservice in the same solution. To do this, you first initialize a `dmpresource` value with the details of the webservice you want to talk by calling `dmpinitwebservice`, then use the `dmphttp` functions to send HTTP requests in a similar way to using plain `mmhttp`. For example, you can send a request to a webservice called **processLoanApps** as follows:

```
declarations
    myservice: dmpresource
end-declarations
dmpinitwebservice(myservice, 'processLoanApps')
if myservice.status <> DMP_OK then
    writeln('Failed to find service due to error:', myservice.lasterror)
    exit(1)
end-if

httpStatusCode := dmphttppost(myservice, '', 'request.json', 'result.json')
if httpStatusCode <> 200 then
    writeln('Error returned by webservice: ', httpStatusCode)
    exit(1)
end-if
```

When calling a webservice, you will typically leave the path field as an empty string.

2.5 Calling DMP Manager

You can also use of the `dmp` module to send HTTP requests to DMP Manager. Initialize a `dmpresource` value with the details of DMP Manager by calling `dmpinitmanager`, then use the `dmphttp` functions to send HTTP requests in a similar way to using plain `mmhttp`. For example, you can send a request to create a commit of the current solution as follows:

```
public declarations
    dmpmanager: dmpresource
    REQUEST_BODY = '{"label":"EXAMPLE","comment":"this is an example"}'
end-declarations
dmpinitmanager(dmpmanager)
if dmpmanager.status <> DMP_OK then
    writeln('Failed to find DMP Manager due to error:', dmpmanager.lasterror)
    exit(1)
end-if

httpStatusCode := dmphttppost(dmpmanager, '/rest/dmp/runtime/solutions/'+getdmpsolid+'/revisions',
```

```

    'text:REQUEST_BODY', 'result.json')
  if httpStatusCode <> 200 then
    writeln('Error returned by DMP Manager: ', httpStatusCode)
    exit(1)
  end-if

```

Note that the set of paths you can call on DMP Manager is restricted as follows:

- You may only make requests to paths pertaining to the solution containing the component executing the model.
- You may make HTTP POST and HTTP GET requests to the `/rest/dmp/runtime/SOLUTIONID/revisions` endpoint.
- You may make HTTP GET requests to the `/rest/dmp/runtime/SOLUTIONID/functions` endpoint.
- You may make HTTP POST and HTTP GET requests to the `/rest/dmp/runtime/SOLUTIONID/services` endpoint.
- You may make HTTP DELETE requests to the `/rest/dmp/runtime/SOLUTIONID/services/SERVICEID` endpoints.
- You may not make any HTTP requests to any other paths.

When calling DMP Manager using the `dmphttp` functions, it is not necessary to start the path with `/com.fico.dmp.manager`.

2.6 Use Outside the Cloud

When not being called from within a DMP Component, the `dmp` module does not give you additional functionality over that supplied by `mmhttp`. However, some users may find it useful to use the `dmp` module outside the cloud when writing libraries to work inside and outside the cloud.

You can use `dmpiniturl` to initialize a DMP resource with a set of user-supplied authorization headers, and then make HTTP requests to that resource using `dmphttp` functions. For example, if you have a local webservice with the URL `http://localhost:8860/` that requires a cookie for authorization, you can call it as follows:

```

declarations
  myurl: dmpresource
  headers: dynamic array(set of string) of text
end-declarations
headers("Cookie") := "SESSIONID=8364825249"
dmpiniturl(myurl, "http://localhost:8860/", headers)
if myurl.status <> DMP_OK then
  writeln('ERROR: ', myurl.lasterror)
  exit(1)
end-if

httpStatusCode := dmphttpget(myurl, '/rest/runtime/status', 'result.json')
if httpStatusCode <> 200 then
  writeln('Error returned from webservice request: ', httpStatusCode)
  exit(1)
end-if

```

2.7 Error handling

You should always check the `status` attribute of the `dmpresource` after every `dmpinit` call, to see if it succeeded or not. This status will be `DMP_OK` on success, `DMP_NOT_FOUND` if the resource you

requested did not exist, `DMP_ACCESS_DENIED` if you don't have access to the resource you requested; any other values indicate internal errors. When the status is not `DMP_OK`, you can find a human-readable error message in the `lasterror` attribute. For example:

```
declarations
  myservice: dmpresource
end-declarations
dmpinitwebservice(myservice, 'processLoanApps')
if myservice.status=DMP_OK then
  writeln('Service found OK')
elif myservice.status=DMP_NOT_FOUND then
  writeln('Service not found')
else then
  writeln('Failed to find service due to error:', myservice.lasterror)
end-if

httpStatusCode := dmphttppost(myservice, '', 'request.json', 'result.json')
if httpStatusCode<>200 then
  writeln('Error returned by webservice: ', httpStatusCode)
  exit(1)
end-if
```

2.8 Limitations

Within an Insight or Executor component, you can only use the `dmp` module from a submodel if the master model also uses the `dmp` module. In the event that your master does not need the `dmp` module, I recommend setting `uses 'dmp'` to the top of it and accessing one of the parameters to prevent the Mosel compiler removing it as an unreferenced module, e.g.

```
uses 'dmp'
setparam('dmp_verbose', false)
```

The `'dmp'` module cannot be used if the model is running locally in an Xpress Workbench component on the cloud. However, the `'dmp'` module will function correctly when Workbench is being used to debug an Insight scenario.

CHAPTER 3

Examples

3.1 Calling a REST endpoint on an Xpress Executor component

This example demonstrates fetching the list of executions from an Xpress Executor component in the same solution.

```
model countexecutions
  uses 'dmp','mmxml'

  declarations
    xecomp: dmpresource
    httpstatus: integer
    doc: xmldoc
    nodes: list of integer
  end-declarations

  ! Initialize dmpresource
  dmpinitcomp(xecomp, 'Xpress Executor')
  if xecomp.status<>DMP_OK then
    writeln('ERROR finding Xpress Executor component: ',xecomp.lasterror)
    exit(1)
  end-if

  ! Make request
  httpstatus := dmphttpget(xecomp, '/rest/runtime/execution', 'executions.json')
  if httpstatus<>200 then
    if xecomp.status<>DMP_OK then
      writeln('ERROR authorizing Xpress Executor component: ',xecomp.lasterror)
    else
      writeln('ERROR returned by Xpress Executor component: ',httpstatus)
    end-if
    exit(1)
  end-if

  ! Use mmxml to parse response
  jsonload(doc, 'executions.json')
  getnodes(doc, '/jsv/jsv', nodes)
  writeln('Executions found: ',nodes.size)

end-model
```

The use of Xpress Executor webservices here is for example purposes only; we recommend using the `executor` module to interact with Xpress Executor components.

3.2 Calling a REST endpoint on DMP Manager

This example demonstrates calling the REST endpoint on DMP Manager to commit a solution revision.

```
model commitsolution
  uses 'dmp','mmsystem'

  declarations
    xecomp: dmpresource
    httpstatus: integer
    public REQUEST_BODY='{"label":"commitSolutionTest","comment":"dmp module example"}'
  end-declarations

  ! Initialize dmpresource
  dmpinitmanager(xecomp)
  if xecomp.status<>DMP_OK then
    writeln('ERROR finding DMP Manager: ',xecomp.lasterror)
    exit(1)
  end-if

  ! Make request
  httpstatus := dmphttppost(xecomp, '/rest/dmp/runtime/solutions/'+getdmprsolid+'/revisions?async=false',
    'text:REQUEST_BODY', 'response.dat')
  if httpstatus<>200 then
    if xecomp.status<>DMP_OK then
      writeln('ERROR making DMP Manager request: ',xecomp.lasterror)
    else
      writeln('ERROR returned by DMP Manager request: ',httpstatus)
    end-if
    exit(1)
  end-if

  writeln('Committed new solution revision')

end-model
```

CHAPTER 4

Types

4.1 The `dmpresource` type

The `dmpresource` type represents a URL and associated authorization credentials. For example, the `dmpresource` initialized by `dmpinitcomp` will contain the URL of the component as well as an appropriate bearer token for authorizing requests. The URL and some other attributes can be read, but the authorization token may not. All the attributes of the `dmpresource` type are read-only and cannot be set directly.

`dmpresource` : type

`resourcetype` : integer

The resource type referred to by the object-the value will correspond to one of the constants listed here:

Values	<code>DMP_RESOURCE_DMP_MANAGER</code>	DMP Manager
	<code>DMP_RESOURCE_COMPONENT</code>	Component
	<code>DMP_RESOURCE_SERVICE</code>	Web service
	<code>DMP_RESOURCE_USER</code>	User-specified URL

`url` : text

The root URL of the DMP resource.

`id` : text

For component resources, this is the component ID. For webservice resources, this is the solution service instance ID. For other resource types, this is unset.

`name` : text

For component resources, this is the component name. For webservice resources, this is the function name. For other resource types, this is unset.

`type` : text

For component resources, this is the component type, e.g. "FICO Xpress Insight". For webservice resources, this is the constant string "SERVERLESS_REST". For other resource types, this is unset.

`env` : string

The DMP lifecycle stage of the resource being referenced; one of the following constants:

Values	<code>DMP_ENV_DESIGN</code>	Design or Root
	<code>DMP_ENV_STAGING</code>	Staging
	<code>DMP_ENV_PRODUCTION</code>	Production

`status` : integer

The status of the last `dmpinit` operation called on this `dmpresource`. One of the following constants:

Values	<code>DMP_OK</code>	Success
	<code>DMP_ACCESS_DENIED</code>	Resource found but you do not have access to it
	<code>DMP_NOT_FOUND</code>	Resource not found
	<code>DMP_IO_ERROR</code>	Internal error reading or writing data
	<code>DMP_PARSE_ERROR</code>	Internal error parsing data

lasterror : text

The error message generated by the last `dmpinit` operation on this `dmpresource`, if it failed.

CHAPTER 5

Procedures and functions

<code>clearerror</code>	Clear error status of dmpresource	p. 11
<code>dmphttpdel</code>	Make HTTP DELETE request to DMP resource	p. 12
<code>dmphttpget</code>	Make HTTP GET request to DMP resource	p. 13
<code>dmphttphead</code>	Make HTTP HEAD request to DMP resource	p. 14
<code>dmphttppost</code>	Make HTTP POST request to DMP resource	p. 15
<code>dmphttpput</code>	Make HTTP PUT request to DMP resource	p. 16
<code>dmpinitcomp</code>	Initialize dmpresource for accessing a DMP component	p. 17
<code>dmpinitmanager</code>	Initialize dmpresource for accessing DMP Manager	p. 18
<code>dmpiniturl</code>	Initialize dmpresource with user-supplied credentials	p. 19
<code>dmpinitwebservice</code>	Initialize dmpresource for accessing a DMP webservice	p. 20
<code>getdmpcompid</code>	Get DMP component ID	p. 21
<code>getdmplifecycleenv</code>	Get DMP component lifecycle	p. 22
<code>getdmpsolid</code>	Get DMP solution ID	p. 23
<code>isdmp</code>	Check if the model is running in DMP	p. 24

clearerror

Purpose

Clears any error in the `dmpresource`, so that the `status` attribute will be `DMP_OK`.

Synopsis

```
procedure clearerror(res:dmpresource)
```

Argument

`res` The `dmpresource`.

Further information

It is not typically necessary to call this procedure, as the error state of the `dmpresource` is cleared automatically on any call to functions that may set it.

dmphttpdel

Purpose

Make an HTTP DELETE request to path within an initialized `dmpresource`.

Synopsis

```
function dmphttpdel(res:dmpresource, path:text, result:text):integer
function dmphttpdel(res:dmpresource, path:text, result:text,
                    xhdr:text):integer
```

Arguments

<code>res</code>	The <code>dmpresource</code> we want to access.
<code>path</code>	The path we want to access within the DMP resource, relative to the resource's root URL
<code>result</code>	File to store the result of the request
<code>xhdr</code>	Additional headers to add to the request

Return value

The HTTP status code of the request, or 999 on a connection error.

Further information

1. This function corresponds to the `httpdel` function of the `mmhttp` module and works in a similar way.
2. The outgoing HTTP request will be automatically augmented with the appropriate credentials for this type of `dmpresource`.
3. In event of an unexpected failure, the request will be retried the number of times specified by `dmp_max_retries` if it returned a HTTP status code listed in `dmp_retry_error_codes`.
4. If this function returns an unexpected HTTP status code, you should check the `status` attribute of the `dmpresource` to see whether this was returned from the HTTP webservice or was the result of an error obtaining credentials.

Example

```
declarations
  res: dmpresource
  httpstatus: integer
  EXEC_ID='5f164857-f27f-44f8-a773-50dedf4f2724'
end-declarations
dmpinitcomp(res,"Xpress Executor")

httpstatus := dmphttpdel(res,'/rest/runtime/execution/'+EXEC_ID,'result.txt')
if httpstatus<>200 then
  if res.status<>DMP_OK then
    writeln('DMP Resource error: ',res.lasterror)
  else
    writeln('Unexpected HTTP status code: ',httpstatus)
  end-if
end-if
```

dmphttpget

Purpose

Make an HTTP GET request to path within an initialized `dmpresource`.

Synopsis

```
function dmphttpget(res:dmpresource, path:text, result:text):integer
function dmphttpget(res:dmpresource, path:text, result:text,
                    xhdr:text):integer
```

Arguments

<code>res</code>	The <code>dmpresource</code> we want to access.
<code>path</code>	The path we want to access within the DMP resource, relative to the resource's root URL
<code>result</code>	File to store the result of the request
<code>xhdr</code>	Additional headers to add to the request

Return value

The HTTP status code of the request, or 999 on a connection error.

Further information

1. This function corresponds to the `httpget` function of the `mmhttp` module and works in a similar way.
2. The outgoing HTTP request will be automatically augmented with the appropriate credentials for this type of `dmpresource`.
3. In event of an unexpected failure, the request will be retried the number of times specified by `dmp_max_retries` if it returned a HTTP status code listed in `dmp_retry_error_codes`.
4. If this function returns an unexpected HTTP status code, you should check the `status` attribute of the `dmpresource` to see whether this was returned from the HTTP webservice or was the result of an error obtaining credentials.

Example

```
declarations
  res: dmpresource
  httpstatus: integer
end-declarations
dmpinitcomp(res, "Xpress Executor")

httpstatus := dmphttpget(res, '/rest/runtime/execution', 'executions.json')
if httpstatus <> 200 then
  if res.status <> DMP_OK then
    writeln('DMP Resource error: ', res.lasterror)
  else
    writeln('Unexpected HTTP status code: ', httpstatus)
  end-if
end-if
```

dmphttphead

Purpose

Make an HTTP HEAD request to path within an initialized dmpresource.

Synopsis

```
function dmphttphead(res:dmpresource, path:text, result:text):integer
function dmphttphead(res:dmpresource, path:text, result:text,
    xhdr:text):integer
```

Arguments

res	The dmpresource we want to access.
path	The path we want to access within the DMP resource, relative to the resource's root URL
result	File to store the result of the request
xhdr	Additional headers to add to the request

Return value

The HTTP status code of the request, or 999 on a connection error.

Further information

1. This function corresponds to the `httphead` function of the `mmhttp` module and works in a similar way.
2. The outgoing HTTP request will be automatically augmented with the appropriate credentials for this type of `dmpresource`.
3. In event of an unexpected failure, the request will be retried the number of times specified by `dmp_max_retries` if it returned a HTTP status code listed in `dmp_retry_error_codes`.
4. If this function returns an unexpected HTTP status code, you should check the `status` attribute of the `dmpresource` to see whether this was returned from the HTTP webservice or was the result of an error obtaining credentials.

Example

```
declarations
    res: dmpresource
    httpstatus: integer
end-declarations
dmpinitcomp(res, "Xpress Executor")

httpstatus := dmphttphead(res, '/rest/runtime/execution', 'null:')
if httpstatus <> 200 then
    if res.status <> DMP_OK then
        writeln('DMP Resource error: ', res.lasterror)
    else
        writeln('Unexpected HTTP status code: ', httpstatus)
    end-if
end-if
```

dmphttppost

Purpose

Make an HTTP POST request to path within an initialized dmpresource.

Synopsis

```
function dmphttppost(res:dmpresource, path:text, data:text,  
                    result:text):integer  
function dmphttppost(res:dmpresource, path:text, data:text, result:text,  
                    xhdr:text):integer
```

Arguments

res	The dmpresource we want to access.
path	The path we want to access within the DMP resource, relative to the resource's root URL
data	Data file to be sent as the request body
result	File to store the result of the request
xhdr	Additional headers to add to the request

Return value

The HTTP status code of the request, or 999 on a connection error.

Further information

1. This function corresponds to the `httppost` function of the `mmhttp` module and works in a similar way.
2. The outgoing HTTP request will be automatically augmented with the appropriate credentials for this type of `dmpresource`.
3. In event of an unexpected failure, the request will be retried the number of times specified by `dmp_max_retries` if it returned a HTTP status code listed in `dmp_retry_error_codes`.
4. If this function returns an unexpected HTTP status code, you should check the `status` attribute of the `dmpresource` to see whether this was returned from the HTTP webservice or was the result of an error obtaining credentials.

Example

```
declarations  
  res: dmpresource  
  httpstatus: integer  
end-declarations  
dmpinitcomp(res, "Xpress Executor")  
  
httpstatus := dmphttppost(res, '/rest/runtime/execution',  
                          'executionrequest.json', 'execution.json')  
if httpstatus<>200 then  
  if res.status<>DMP_OK then  
    writeln('DMP Resource error: ', res.lasterror)  
  else  
    writeln('Unexpected HTTP status code: ', httpstatus)  
  end-if  
end-if
```

dmphttpput

Purpose

Make an HTTP PUT request to path within an initialized dmpresource.

Synopsis

```
function dmphttpput(res:dmpresource, path:text, data:text,  
    result:text):integer  
function dmphttpput(res:dmpresource, path:text, data:text, result:text,  
    xhdr:text):integer
```

Arguments

res	The dmpresource we want to access.
path	The path we want to access within the DMP resource, relative to the resource's root URL
data	Data file to be sent as the request body
result	File to store the result of the request
xhdr	Additional headers to add to the request

Return value

The HTTP status code of the request, or 999 on a connection error.

Further information

1. This function corresponds to the `httpput` function of the `mmhttp` module and works in a similar way.
2. The outgoing HTTP request will be automatically augmented with the appropriate credentials for this type of `dmpresource`.
3. In event of an unexpected failure, the request will be retried the number of times specified by `dmp_max_retries` if it returned a HTTP status code listed in `dmp_retry_error_codes`.
4. If this function returns an unexpected HTTP status code, you should check the `status` attribute of the `dmpresource` to see whether this was returned from the HTTP webservice or was the result of an error obtaining credentials.

Example

```
declarations  
    res: dmpresource  
    httpstatus: integer  
end-declarations  
dmpinitcomp(res, "Example Component")  
  
httpstatus := dmphttpput(res, '/rest/design/model',  
    'modelbody.dat', 'response.dat')  
if httpstatus<>200 then  
    if res.status<>DMP_OK then  
        writeln('DMP Resource error: ', res.lasterror)  
    else  
        writeln('Unexpected HTTP status code: ', httpstatus)  
    end-if  
end-if
```

dmpinitcomp

Purpose

Initialize a `dmpresource` to access a DMP component.

Synopsis

```
procedure dmpinitcomp(res:dmpresource, type:text)
procedure dmpinitcomp(res:dmpresource, id:text, type:text, name:text,
                      env:text)
```

Arguments

<code>res</code>	The <code>dmpresource</code> value to initialize.
<code>id</code>	The ID of the component you want to access (or empty string)
<code>type</code>	The type of the component you want to access (or empty string)
<code>name</code>	The name of the component you want to access (or empty string)
<code>env</code>	The lifecycle stage of the component instance you want to access (or empty string), one of: DMP_ENV_DESIGN Design or Root DMP_ENV_STAGING Staging DMP_ENV_PRODUCTION Production

Further information

1. This procedure will initialize a `dmpresource` to access another component in the same solution as the component executing the model.
2. You can pass an empty string if you don't want to specify any of the arguments, but you must specify at least one of `id`, `type` and `name`.
3. If you specify a component type and a component of this type does not exist in the solution, this function will look for a component whose type contains the type string you specified (e.g. passing type *Xpress Insight* may return a component of type *Xpress Insight 4.12* if one exists).
4. Where multiple components in the solution match the values you specified, the `dmpresource` will be initialized to access an arbitrarily selected component.
5. If you don't specify a lifecycle environment, the lifecycle of the component executing the model will be used.
6. You should check the `status` attribute of the `dmpresource` after calling this procedure.

Example

```
declarations
  res: dmpresource
end-declarations
dmpinitcomp(res, "Xpress Executor")
if res.status<>DMP_OK then
  writeln('ERROR: ', res.lasterror)
end-if
```

dmpinitmanager

Purpose

Initialize a `dmpresource` to access DMP Manager.

Synopsis

```
procedure dmpinitmanager(res:dmpresource)
```

Argument

`res` The `dmpresource` value to initialize.

Further information

You should check the `status` attribute of the `dmpresource` after calling this procedure.

Example

```
declarations
  res: dmpresource
end-declarations
dmpinitmanager(res)
if res.status<>DMP_OK then
  writeln('ERROR: ',res.lasterror)
end-if
```

dmpiniturl

Purpose

Initialize a `dmpresource` with user-supplied URL and credentials

Synopsis

```
procedure dmpiniturl(res:dmpresource, url:text, authtokentype:text,  
                    authtoken:text)  
procedure dmpiniturl(res:dmpresource, url:text, authheaders:array(set of  
                    string) of text)
```

Arguments

<code>res</code>	The <code>dmpresource</code> value to initialize.
<code>url</code>	The resource root URL. All <code>dmphttp</code> calls through this resource will be relative to this URL.
<code>authtokentype</code>	The type of authorization token, expressed as one of the following constants: DMP_TOKEN_TYPE_BEARER DMP_TOKEN_TYPE_JWT
<code>authtoken</code>	The authorization token to add to outgoing HTTP requests made through this resource.
<code>authheaders</code>	A collection of HTTP Headers to add to outgoing HTTP requests made through this resource.

Further information

1. This procedure is intended to assist users to write code interacting with DMP resources that will work both within Xpress DMP components and when executed locally.
2. If you pass an `authtoken` value, all HTTP requests made through this resource will be decorated with an appropriate `Authorization:` header.
3. If you pass an `authheaders` value, all HTTP requests made through this resource will be decorated with HTTP headers read from this array (indices are the header names).
4. You should check the `status` attribute of the `dmpresource` after calling this procedure.

Example

```
declarations  
  res: dmpresource  
  headers: dynamic array(set of string) of text  
end-declarations  
headers("Cookie") := "SESSIONID=8364825249"  
headers("Authorization") := "Bearer 834692364jdnvdjfvb7t3jkd78"  
dmpiniturl(res, "http://fakeresource.example.com/", headers)  
if res.status<>DMP_OK then  
  writeln('ERROR: ',res.lasterror)  
end-if
```

dmpinitwebservice

Purpose

Initialize a `dmpresource` to access a DMP webservice.

Synopsis

```
procedure dmpinitwebservice(res:dmpresource, functionid:text)
procedure dmpinitwebservice(res:dmpresource, id:text, functionid:text,
                             env:text)
```

Arguments

<code>res</code>	The <code>dmpresource</code> value to initialize.
<code>id</code>	The ID of the solution service you want to access (or empty string)
<code>functionid</code>	The ID of the function you want to access, ie the function name (or an empty string)
<code>env</code>	The lifecycle stage of the service you want to access (or empty string), one of: DMP_ENV_DESIGN Design or Root DMP_ENV_STAGING Staging DMP_ENV_PRODUCTION Production

Further information

1. This procedure will initialize a `dmpresource` to access a DMP function deployed as a `SERVERLESS_REST` type solution service.
2. You can pass an empty string if you don't want to specify any of the arguments, but you must specify at least one of `id` and `functionid`.
3. The `functionid` value is usually the function name.
4. If you don't specify a lifecycle environment, the lifecycle of the component executing the model will be used.
5. You should check the `status` attribute of the `dmpresource` after calling this procedure.

Example

```
declarations
  res: dmpresource
end-declarations
dmpinitwebservice(res, "getBestPolicy")
if res.status<>DMP_OK then
  writeln('ERROR: ', res.lasterror)
end-if
```

getdmpcompid

Purpose

Gets the ID of the DMP component executing this model.

Synopsis

```
function getdmpcompid:text
```

Return value

The component ID.

Further information

This function returns the component ID (the same for design, staging and production lifecycle environments) and not the component instance ID (which is different for each environment).

getdmplifecycleenv

Purpose

Gets the DMP lifecycle environment of the component instance executing this model.

Synopsis

```
function getdmplifecycleenv:string
```

Return value

One of the constant values DMP_ENV_DESIGN, DMP_ENV_STAGING and DMP_ENV_PRODUCTION.

getdmpsolid

Purpose

Gets the ID of the DMP solution to which the component executing this model belongs.

Synopsis

```
function getdmpsolid:text
```

Return value

The solution ID.

isdmp

Purpose

Checks if the model is running within a DMP component that supports the `dmp` Mosel module.

Synopsis

```
function isdmp:boolean
```

Return value

`true` if running within DMP, `false` if not.

Further information

If this returns `false`, then any other functions you call from this module will generate errors.

CHAPTER 6

Parameters

Via the `getparam` function and the `setparam` procedure it is possible to access the following control parameters of module `dmp` :

<code>dmp_max_retries</code>	Number of times to retry failed requests	p. 25
<code>dmp_retry_error_codes</code>	Types of error code to retry	p. 26
<code>dmp_verbose</code>	Activate additional logging	p. 25

`dmp_verbose`

Description	When set to <code>true</code> , the module will produce additional logging output to the model's error stream.
Type	Boolean, read/write
Default value	<code>false</code>
Note	This can be useful for diagnosing errors.

`dmp_max_retries`

Description	Set the maximum number of times a failed HTTP request will be retried.
Type	Integer, read/write
Default value	8
Note	Each retry will be after an exponentially larger delay (200ms, 400ms, 800ms, etc) so be aware that large values can take a long time to report errors.
Note	The default setting will retry for approximately 52 seconds.
Note	The types of operation that are retried can be controlled using the <code>dmp_retry_error_codes</code> parameter.

dmp_retry_error_codes

Description	When a <code>dmphttp</code> request returns an HTTP status code found in this comma-separated list, it will be retried, up to the number of times specified by <code>dmp_max_retries</code> .
Type	String, read/write
Default value	999, 500, 504, 401

APPENDIX A

Contacting FICO

FICO provides clients with support and services for all our products. Refer to the following sections for more information.

Product support

FICO offers technical support and services ranging from self-help tools to direct assistance with a FICO technical support engineer. Support is available to all clients who have purchased a FICO product and have an active support or maintenance contract. You can find support contact information and a link to the Customer Self Service Portal (online support) on the Product Support home page (www.fico.com/en/product-support).

The FICO Customer Self Service Portal is a secure web portal that is available 24 hours a day, 7 days a week from the Product Support home page. The portal allows you to open, review, update, and close cases, as well as find solutions to common problems in the FICO Knowledge Base.

Please include 'Xpress' in the subject line of your support queries.

Product education

FICO Product Education is the principal provider of product training for our clients and partners. Product Education offers instructor-led classroom courses, web-based training, seminars, and training tools for both new user enablement and ongoing performance support. For additional information, visit the Product Education homepage at www.fico.com/en/product-training or email producteducation@fico.com.

Product documentation

FICO continually looks for new ways to improve and enhance the value of the products and services we provide. If you have comments or suggestions regarding how we can improve this documentation, let us know by sending your suggestions to techpubs@fico.com.

Please include your contact information (name, company, email address, and optionally, your phone number) so we may reach you if we have questions.

Sales and maintenance

If you need information on other Xpress Optimization products, or you need to discuss maintenance contracts or other sales-related items, contact FICO by:

- Phone: +1 (408) 535-1500 or +44 207 940 8718
- Web: <http://www.fico.com/optimization> and use the available contact forms

Related services

Strategy Consulting: Included in your contract with FICO may be a specified amount of consulting time to assist you in using FICO Optimization Modeler to meet your business needs. Additional consulting time can be arranged by contract.

Conferences and Seminars: FICO offers conferences and seminars on our products and services. For announcements concerning these events, go to www.fico.com or contact your FICO account representative.

FICO Community

The FICO Community is a great resource to find the experts and information you need to collaborate, support your business, and solve common business challenges. You can get informal technical support, build relationships with local and remote professionals, and improve your business practices. For additional information, visit the FICO Community (community.fico.com/welcome).

About FICO

FICO (NYSE:FICO) powers decisions that help people and businesses around the world prosper. Founded in 1956 and based in Silicon Valley, the company is a pioneer in the use of predictive analytics and data science to improve operational decisions. FICO holds more than 165 US and foreign patents on technologies that increase profitability, customer satisfaction, and growth for businesses in financial services, telecommunications, health care, retail, and many other industries. Using FICO solutions, businesses in more than 100 countries do everything from protecting 2.6 billion payment cards from fraud, to helping people get credit, to ensuring that millions of airplanes and rental cars are in the right place at the right time. Learn more at www.fico.com.

Index

A

AWS Lambda, 3

C

clearerror, 11

D

DMP Component, 2

DMP Manager, 3, 6

DMP Webservice, 3

dmp_max_retries, 25

dmp_retry_error_codes, 26

dmp_verbose, 25

dmphttpdel, 2

dmphttpdel, 12

dmphttpget, 2, 6

dmphttpget, 13

dmphttphead, 2

dmphttphead, 14

dmphttppost, 2, 6

dmphttppost, 15

dmphttpput, 2

dmphttpput, 16

dmpinitcomp, 2

dmpinitcomp, 17

dmpinitmanager, 18

dmpiniturl, 4

dmpiniturl, 19

dmpinitwebservice, 3

dmpinitwebservice, 20

dmpresource, 2-4, 6, 8, 12-20

dmpresource, 8

E

env, 8

G

getdmpcompid, 2

getdmpcompid, 21

getdmplifecycleenv, 2

getdmplifecycleenv, 22

getdmpsolid, 2

getdmpsolid, 23

I

isdmp, 2

isdmp, 24

L

lasterror, 4, 8

N

name, 8

R

resourcetype, 8

S

status, 4, 8

submodel, 5

T

type, 8

U

url, 8

X

Xpress Executor, 1, 6

Xpress Insight, 1

Xpress Workbench, 5