GRAPHISOFT Technote

BIMx API
v4.2

August 2020

Gyuri Nyitrai
BIMx Product Manager
gynyitrai@graphisoft.com

Table of Contents

Introduction

The primary goals of the BIMx API are to make available certain BIMx mobile app functions for construction IT and FM solution providers and to enrich their solutions by providing them BIM model and documentation viewer functionality. BIMx API also a viable solution for enterprises to integrate BIMx mobile apps into their standard environment and workflows.

The purpose of this document is to describe the capabilities of the BIMx API functionality, especially the concept of the BIMx Extensions and how to develop them.

This API consists several methods to make accessible BIMx app's core functionality for 3rd party developers and extendible its content from external sources. Some of its functions can be accessed anytime by sending custom URLs with appropriate parameters, others required to install your extension definition (.json) file beforehand.

The BIMx API is available on iOS and Android platforms alike.

To run a BIMx API function you must possess valid BIMx Developer ID issued by GRAPHISOFT and BIMx PRO license on your mobile devices.
Note: BIMcloud User License doesn't enable BIMx API functions.

What's new in the v4.2 version

- Updated URL parameter names
- Optional encoding of your extension .json files
- Updated demo extension site link

BIMx Extensions

BIMx Extension is a basic text file in simple .json (or its encoded) format, which contains your customization options as key – value pairs.

With BIMx Extensions, you can:

- Add new custom menu items to the context menu of any selected 3D model element(s) or to the “More options” menu (also known as the “...” menu) of the BIMx apps. By tapping on those new menu items, you can either open a webpage or another app on the mobile device. You can pass information over about the current Hyper-model or the highlighted element to the webpage or to the other app.
- Display additional information related to the current Hyper-model or to the highlighted element. The additional information should be provided by your web service. You can also pass information to your web service about the current Hyper-model and the highlighted element when requesting the additional data to be displayed.

Below you can find the details of each of four extension points together with explanatory sample extensions.

Developing and testing extensions

Extensions are described in text files (in JSON format). You can create and edit extensions with any text editor program. The file name extension of those files should be “.bimxx” (BIMx eXtension).

Once an extension file has been created it has to be installed in BIMx. It can be installed either by opening the .bimxx file in the BIMx app (e.g. by tapping on a .bimxx file attached to an email sent to your iPad) or by tapping on a specially formatted link on a webpage (for example *“bimxapplication://installExtension?url=https%3a%2f%2fwww.dropbox.com%2fs%2fv7h637q7h3lpkay%2ftest1.bmxx%3fdl%3d0%26raw%3d1”,* where the “url” parameter contains the URL of the extension file). Note that you can open such links from your app, too, meaning that you can add for example a button to your app which installs your extension in BIMx.

Installed extensions can be updated both manually or automatically (if and only if the extension contains a link to a website from where the new version of the extension could be downloaded):

- Manual update: open a Hyper-model in BIMx, open its 3D model, then open the “...” (More options) menu, select Settings and look for a section titled “Installed extensions” (iOS) or tap on “Installed extensions” (Android). You will see all your installed extensions listed there. Tapping on the extension’s name will display a menu of extension-related operations (reload and delete). Note that you have to long tap on Android to have the menu displayed.
- Automatic update: every time you start BIMx (either after restarting your mobile device or after quitting the app), extensions will automatically be updated.

The easiest way to develop an extension is to store the .bimxx file in e.g. Dropbox, edit the file using your text editor and manually update the extension in BIMx after you have made the necessary changes.

Sample extension file

```
{
 "name": "Sample extension",
 "developer_id": "<INSERT YOUR BIMX DEV ID HERE>",
 "update_url":
"https://www.dropbox.com/s/kwkqvlq6t85e60s/MyExtension.bimxx?dl=0&ra
w=1",
 "hyper_model_names": [
   "BIMx Demo Hyper-model"
 ],
 "elem_context_menu": [
   {
     "title": "Open Sandbox app",
     "url":
"sandbox://open?element_guid=$(element_guid)&hyper_model_name=$(hype
r_model_name)",
     "position": "left"
   }
 ],
 "project_info": [
   {
     "title": "Open Graphisoft website",
     "url": "http://www.graphisoft.com"
   }
 ]
}
```

This sample extension adds a new item to the element context menu. The new menu item is shown as “Open Sandbox app” and opens an app called “Sandbox” passing the GUID of the highlighted element plus the name of the current Hyper-model to the app. It also adds an item to the “...” menu. The item is shown as “Open Graphisoft website” and opens the device’s default web browser showing Graphisoft’s website.

Extension structure

Extension files have the following main items:

- Name (“name”, string, required): the name of the extension. Please make sure the name is unique (i.e. there are no two extensions with the same name installed)
- Developer ID (“developer_id”, string, required): a GUID identifying the extension’s developer. This is required to ensure only authorized developers can create and distribute extensions. Please contact Graphisoft to get your developer ID. See the “Developer ID” section for further details. Note the developer ID in the above sample file is not valid and will not work.
- Update URL (“update_url”, string, optional): the URL from where the extension could be updated. The file could either be stored on a web server or in a cloud file storage. If you store the extension file in Dropbox, you can get an URL what you can use as the update URL: locate the file in Explorer (Windows) or Finder (Mac), right-click on it and select “Copy Dropbox Link”. Paste the URL to your extension file (see the sample

above) and make sure to add “&raw=1” to the end of the URL, otherwise updating won’t work

- Hyper-model names (“hyper_model_names”, array of strings, optional): you may want an extension to be available only in certain hyper-models. Optionally you can specify a list of Hyper-model names in the extension file. If you do so, the extension will remain hidden and disabled unless the name of the Hyper-model matches (case-insensitive) one of the names specified in the extensions file.
- Selected element context menu (“elem_context_menu”, array of dictionaries, optional): it describes the menu items added to the element context menu. See details below.
- Project info (“project_info”, array of dictionaries, optional): it describes the menu items added to the “...” (also known as “More options”) menu. See details below.
- Element Info (“elem_info”, dictionary, optional): it describes the details of the web service providing extra data in element info. See details below.
- Model info (“model_info”, dictionary, optional): it describes the details of the web service providing extra data in model info dialog. See details below.

Variables in URLs

You can use the following variables in URLs:

- Element related variables:
  Note: in the cases when there is no element selected (Project Info or Model Info), then these variable names will be replaced with an empty string.
  - $(element_guid): it will be replaced by the GUID of the highlighted 3D model element
  - $(selected_element_guids): it will be replaced by a comma-separated list of element GUIDs of the selected elements
  - $(elem_id): it will be replaced by value (optionally) provided in the ARCHICAD Settings dialog for the selected model element; it could only be used in URLs extending the element context menu and button type items in the Element Info popover; in other cases, or if not available in the 3D model or if the highlighted element is a zone, then it will be replaced by an empty string; see ARCHICAD Users’ Guide for more information on Element IDs.
  - $(zone_id): it will be replaced by the ID of the selected Zone; it could only be used in URLs extending the element context menu and button type items in the Element Info popover; in other cases, or if the highlighted element is not a Zone, it will be replaced by an empty string; see ARCHICAD Users’ Guide for more information on Zone IDs.
- Model related variables:
  - $(hyper_model_name): name of the current Hyper-model; it could be used in any URL
  - $(project_id): it will be replaced by the project ID of the current Hyper-model; it could be used in any URL
  - $(project_name): it will be replaced by the project name of the current Hyper-model; it could be used in any URL

- $(first_measure_point): if the current view is the 3D view and the measure tool is active, then it will be replaced by the x,y,z coordinates of the first measure point. Otherwise it will be replaced by an empty string (“”). Example value: 14.2731,0.7517,8.0763
- $(camera_pos): if the current view is the 3D view, then it will be replaced by the x,y,z coordinates of the camera. Otherwise it will be replaced by an empty string (“”).
- $(view_direction): if the current view is the 3D view, then it will be replaced by the x,y,z coordinates of the camera’s view direction vector. Otherwise it will be replaced by an empty string (“”). The vector is normalized, i.e. it’s length is always 1.

BIMx Developer ID

A developer ID (a GUID) has to be included in the extension file to make it work (please contact Graphisoft to get your developer ID if you haven’t done so yet). If the developer ID is missing from the file, the extension will be installed but will remain disabled and its additions to BIMx will not show up/work. A warning message (“disabled; devID missing”) will be shown next to the extension’s name in the list of installed extensions.

In case of unauthorized use developer IDs can be revoked by Graphisoft. Every time an extension is loaded (at installation and at BIMx start-up), the developer ID included in the extension file is checked online. If the developer ID is invalid or has been revoked, the extension will be disabled. In such cases a warning message (“disabled”) will be shown next to the extension’s name in the list of installed extensions.

Sometimes the developer ID can’t be checked online for some reason (e.g. the user is offline). To make sure the extension remains operational in such cases, there is a 7-days long “grace period”: the extension will remain enabled and operation for 7-days since the last successful check of the developer ID. Note that this grace period is not applicable to missing or revoked developer IDs. Extensions without a developer ID will be disabled immediately. Extensions with a revoked developer ID will be disabled as soon as the developer ID authentication server gets available again.

Option to encode your extension

To protect your privacy Graphisoft offers you an option to encode your custom extensions by providing command line tools both for Windows and macOS. You can download the tools from here: https://graphisoft.sharefile.com/d-sed893507903548b2b3df797a859b28a6

While it is easier to handle the raw json extension file format during the development and testing phase, it is suggested to release your extension in encoded form to avoid revealing your developer id or urls.

To use the encoder command line tool open a Command Prompt (Windows) or a Terminal window (macOS) and enter:

bxxencoder.exe <path to .bimxx file> (on Windows)

bxxencoder <path to .bimxx file> (on macOS)

The tool will create a new file beside the original, appending -encrypted to it's name. You may rename it as you like leaving the filename extension (bimxx) unchanged.

Windows example:

> *C:\>D:\Dev\BIMxMain\Tools\bxxencoder\bin\win\bxxencoder.exe D:\Dev\BIMxExt\myextension.bimxx*
>
> *Processing D:\Dev\BIMxExt\myextension.bimxx...*
>
> *Done, D:\Dev\BIMxExt\myextension-encrypted.bimxx has been created*
>
> *C:\>*

macOS example:

> *meX13:~ me$ /Volumes/WK/BIMxMain/Tools/bxxencoder/bin/osx/bxxencoder /Users/me/Desktop/myextension.bimxx*
>
> *Processing /Users/me/Desktop/myextension.bimxx...*
>
> *Done, /Users/me/Desktop/myextension-encrypted.bimxx has been created*
>
> *meX13:~ me$*

Authentication of web service requests

Optionally you can require HTTP basic authentication for the “Element info list” and “Model info dialog” related web service requests (see details above) in order to avoid unauthorized access to your data available via the web service(s).

If the “authentication”: “basic” key-value pair is present in your extension (see above), then BIMx will ask the user to enter his/her username and password when the extension is installed. If the user doesn’t enter the required credentials or the entered credentials aren’t valid (e.g. the password has changed), then BIMx will ask for them again “on-demand” (i.e. when the web service request is made). Your web service should respond with either 401 (unauthorized) or 403 (forbidden) status codes if the credentials are not present or invalid (the rest of the response will be ignored in this case). Those status codes will cause BIMx to ask for the credentials.

Username and password values are securely stored on the user’s device until the user logs out or the extension gets deleted (see below).

Web services are identified by their host name, i.e. BIMx will use the same username and password both for the “element info” and the “model info” web services if the hostname part of their URL is the same. For security reasons username and password values are **not shared** between extensions, i.e. even if two different extensions refer to the very same web service, the user will have to enter the credentials for both extensions separately.

If the user wishes to remove the previously entered username and password from his/her device, then he/she should find the extension in “Extension Manager” and select “Logout” from the extension’s context menu (available only if credentials have been entered before).

When the extension is deleted (uninstalled), corresponding usernames and passwords are also deleted.

When the extension is updated (either manually or automatically), previously entered credentials will be preserved.

Extension points

Below you can find detailed explanations for each four currently available user interface points in BIMX app where you can introduce your business logic.

Element context menu

You could add one or more new buttons to the context menu shown in the screenshot. You could specify the items' location, text and URL to open when you tap the button. This extension point could be used to for example display a video stream in a popover window or open the device's browser to create a maintenance ticket for the selected element. You could specify which elements should the new button be shown for (see "filtering" below).

Specification

Additions to the element context menu are described in the (optional) "elem_context_menu" array of the extension file. Each item in the array describes one new button. Item attributes:

- Title (string; required): text to be shown in the context menu
- URL (string; required): the URL to be opened when the user taps the button; it might contain variable names; those will be substituted prior to opening the URL; variable values will be URL (percent) encoded; both HTTP (http:// and https://) and custom URL schemes (e.g. yourapp://) are supported and the later could be used to transfer data/perform a function of an external mobile app.
- Position (string; optional; allowed values are "left" and "right"; default: "right"): it defines if the button should be shown at the beginning or at the end of the context menu; buttons defined in a single extension will be added to the context menu in the order they appear in the extension (considering their "position" value); the order of button "groups" defined in separate extensions is undefined. Note that on Android you can have **5 or less items on each side** (all extensions combined) for the time being
- Style (string; optional; allowed values are "popover", "modal" and "browser"; default: "modal"): it defines if the URL should be opened in a popover, in a modal window in the BIMx app or in the device's default browser app; in case of non-HTTP URLs (e.g. yourapp://) the value of style will be ignored, iOS will switch to the app defined by the custom URL scheme. Note that "popover" and "modal" styles on iOS are supported **only on iOS 9 or newer**.
- Filtering ("shown_only_if"; optional; default: context menu item is shown for all elements): if present, it defines when to show the context menu item. The context menu will be shown if at least one criteria is met (i.e. there is an "OR" connection between the criteria):
    - Element GUID ("element_guids"; optional, array of strings): the context menu item will be shown if the element GUID of the selected element matches one of

the element GUIDs in the array; note that your extension will be Hyper-model specific if this criterion is used

- Metadata: ("metadata"; optional): you can refer to either the presence or the value of certain exported info items:
  - "is_present" (optional, array of strings): if there is an element info item where "key" is the same (comparison is case-insensitive) as one of the values provided in this string, then the menu item will be shown
  - "equals" (optional): if there is a key-value pair in the element info where both the key and the value matches one of the key-value pairs provided in this array, then the menu item will be shown

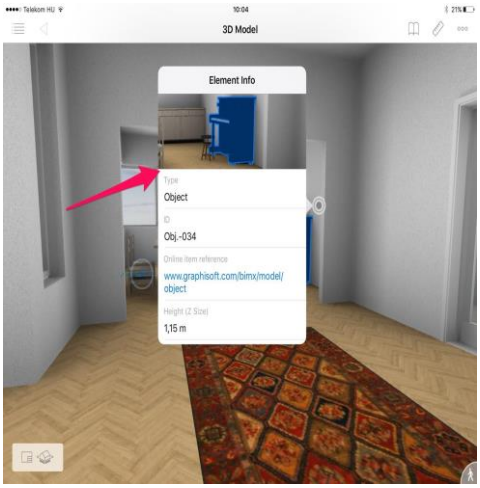
- Upload the content of the 3D view as a JPEG image ("upload_image"; optional, default: don't upload): optionally you can define a location where a JPEG image will be uploaded to prior to opening the URL defined by the "url" value above. BIMx will upload a portion of the 3D view containing the highlighted element and its surrounding (bounding box of the element + 25% of the bounding box in each direction). Details to be specified:
  - "url" (required, string): an HTTP(S) link defining the upload target; the URL might contain any applicable variable
  - "method" (optional, string, default value: "POST"): the HTTP method to be used for uploading; usually it's either POST or PATCH
  - "authentication" (optional, string, default value: no authentication is required): you can optionally specify the value "basic". If you do so, BIMx will ask for a username and a password before uploading the image. BIMx will store the entered credentials and will reuse them for later uploads. The entered values will be sent as a standard "Authentication" HTTP header
  - BIMx will send an HTTP request where the value of "Content-Type" is set to "multipart/form-data". HTTP method will be set as defined by the "method" value. The HTTP request's body will contain two parameter values (in a format specified by RFC1521):
    - "element_guid": UTF-8 encoded lowercase value of the highlighted element's GUID
    - "screenshot": a binary representation of a JPEG image
  - During the image upload BIMx will show an activity indicator. The user might cancel the image upload by tapping on "Cancel" (iOS) or by tapping the hardware "back" button or tapping on the background (Android). If the user cancels the upload, the original (content) URL won't be opened either.

Example code snippet

```
…
"elem_context_menu": [
 {
   "title": "Show specs",
   "url": "https://www.yourdomain.com/specs/$(element_guid)",
   "position": "left",
```

```
    "style": "browser"
  },
  {
    "title": "Camera feed",
    "url": "https://www.feeds.com/feed?camera_id=$(element_guid)",
    "style": "popover",
    "shown_only_if": {
      "element_guids": [
        "5E258C98-C74E-43E6-9F8A-894F2F9A3E22",
        "463F5328-53EC-4255-AC9F-614AB6B26C10"
      ],
      "metadata": {
        "is_present": [
          "Camera location"
        ],
        "equals": [
          {
            "key": "Type",
            "value": "Camera 1"
          }
        ]
      }
    }
  },
  {
    "title": "Report issue",
    "url”: "yourapp://report?id=$(element_guid)",
    "position": "left",
    "upload_image": {
      "url":
"https://example.com/upload_image?element_guid=$(element_guid)",
      "method": "POST",
      "authentication": "basic"
    }
  }
]
…
```

Element Info list



You could add new items to the Element Info popover. Data to be displayed should be provided by a web service. You have to specify the web service's URL (only HTTP(S) GET method is supported) to fetch data from. Response has to be a JSON file (see format specification below). The JSON file has to contain the title (key) and the value of the added items.

Specification

Details of your web service providing additional items for the Element Info popover are described in the (optional) "elem_info" item of the extension file:

- URL (string; required): when end user opens the element info popover, BIMx sends a HTTP(S) GET request to the URL provided, waits for the response (a JSON file, see specification below), parses the response and adds items to the Element Info popover based on the response. The URL might contain variable names; those will be substituted prior to sending the request. Variable values will be URL (percent) encoded. Please note that only HTTP (http:// and https://) URLs are supported for the time being. The HTTP request will be cancelled if the user closes the Element Info popover before the response is received. Currently only HTTP basic authentication (other than HTTP request header fields, see below) is supported, i.e. the request should either work without any further authentication challenge or require the presence of the "Authorization" HTTP request header field with "Basic <Base64 encoded username:password>" value. If the HTTP request fails for some reason (non-2xx status code, timeout, etc.), then a brief error message will be displayed for the user unless it's authentication related (see below).
- HTTP header fields (array of strings, optional): in order to protect data provided by the web service and/or identify the app making the request, you could define custom HTTP header field values (e.g. "Authorization: key=<your_authorization_key>")
- Authentication (string, optional): the only accepted value is "basic", any other value will be ignored. See details below.

Example code snippet

```
…
"elem_info": {
 "url":
"https://www.yourwebservice.com/resource?id=$(element_guid)",
```

```
 "http_headers": [
   {
     "field": "Accept",
     "value": "application/json"
   },
   {
     "field": "Authorization",
     "value": "key=hG7k6ZG8bsd6gHK89a"
   }
 ],
 "authentication": "basic"
}
…
```

Response JSON format specification

The JSON file received as the response from the web service should be encoded using UTF-8 encoding. It should contain an array of items. (If the response is empty, then no item will be added to the Element Info popover.) For each item, you should provide the following details:

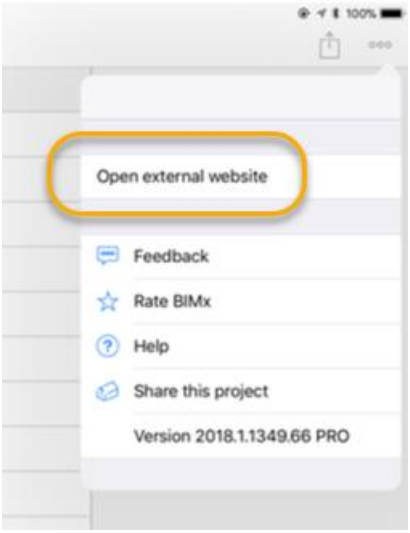
- Key (string, optional): the title to be shown as the first line of text in the item; formatting (font, colour, alignment, etc.) will match the formatting of the items “built-in” to BIMx. If “key” is not provided or empty, then only one line of text will be shown in the Element Info popover
- Value (string, required): the value to be shown as the second line of text in the item; formatting (font, colour, alignment, etc.) will match the formatting of the items “built-in” to BIMx (i.e. links will be shown with blue). If it starts with a “[“ character, then BIMx will try to parse it: the string in square brackets will be used as anchor text and shown as “value” in the BIMx user interface; the string in parenthesis will be used as URL and will be opened if the user taps on this item.
- Please note that if you wish to display an URL as a Value then you can use markdown syntax to hide the details of your URL. You can use it in the following form: [text](url)
  For example:
  `[This is the display text of the link]`
  `(https://www.ThisIsTheActualSite.com:8080/bimx?`
  `param1=with+several&param2=parameters)`
- Position (string; optional; allowed values are “top” and “bottom”; default: “bottom”): it defines where the new item should show up (“top” means above the element’s built-in properties, “bottom” means below the built-in properties); the order of item in the UI will match the order of items (with the same position) in the JSON file, but the order of the items from different web services is undefined

Example response JSON

```
"items": [
 {
   "key": "FM ID",
   "value": "769876",
   "position": "top"
 },
 {
   "key": "File a report",
   "value": "https://www.fmdomain.com/report?id=$(element_guid)"
```

```
  },
  {
    "key": "View datasheet",
    "value": "[Tap here to view the
datasheet](https://www.fmdomain.com/datasheet?id=$(element_guid))",
  },
  {
    "value": "[Show in FM app](fmapp://show?id=$(element_guid))"
  }
]
```

Project Info menu



You could add one or more items to the project info menu. The items have to be specified in the “project_info” item (optional, array of items) of your extension file. Position is fixed (i.e. any “position” value will be ignored). All new items will be inserted below the “Settings” item (as noted on the screenshot above). The order of the new items will be the same as in the extension file. The order of items from different extensions is undefined. Extension items will have an unalterable “puzzle” icon in the menu. On iOS extension defined menu items will be in a separate group (the same group for all extensions). Note that on Android you can have **10 or less extension items** (all extensions combined) in the Project Info menu for the time being.

Specification

Additions to the Project Info menu are described in the (optional) “project_info” array of the extension file. Each item in the array describes one menu item. Item attributes:

- Title (string; required): text to be shown in the project info menu (the item will have no icon)
- URL (string; required): the URL to be opened when the user taps the item; it might contain variable name; those will be substituted prior to opening the URL; variable values will be URL (percent) encoded; both HTTP (http:// and https://) and custom URL schemes (e.g. yourapp://) are supported and the later could be used to transfer data/perform a function of an app other than BIMx
- Style (string; optional; allowed values are “modal” and “browser”; default: “modal”): it defines if the URL should be opened in a modal window within the BIMx app or in the

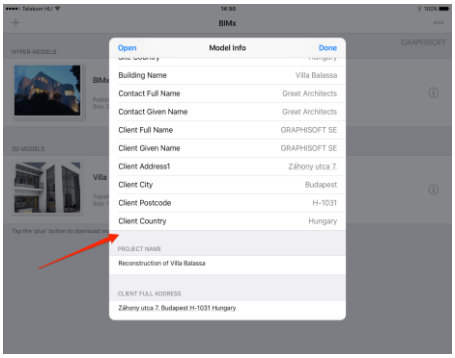
device's default browser app; in case of non-HTTP URLs (e.g. yourapp://) the value of style will be ignored; iOS will switch to the app defined by the custom URL scheme.

- Upload the content of the 3D view as a JPEG image: it's the same as described in the "Element context menu" section with two exceptions (see further details there):
    - In this case the uploaded JPEG image will contain the entire 3D view. The resolution of the uploaded image depends on the orientation and the resolution of the device BIMx is running on. BIMx won't scale the image before uploading, so in case of high resolution ("Retina") devices, the image will also be a high-resolution one.
    - The HTTP request's body will contain "hyper_model_name" instead of "element_guid"

Example code snippet

```
…
"project_info": [
 {
   "title": "View project specs",
   "url":
"https://www.yourdomain.com/projectspecs?id=$(project_id)",
   "style": "browser"
 },
 {
   "title": "Post project note",
   "url": "https://www.yourdomain.com/projectnote?id=$(project_id)",
   "style": "modal"
 },
 {
   "title": "Import project to FM app",
   "url": "yourapp://import?id=$(project_id)",
   "upload_image": {
     "url": "https://example.com/upload_image",
     "method": "POST",
     "authentication": "basic"
   }
 }
]
…
```

Model Info dialog



Similar to the extension of the Element Info popover, you could add items to the Model Info dialog. The details of these additions have to be provided by a web service. The details of the web service (URL, optional HTTP header fields) are described in the optional “model_info” item of your extension.

Specification

Details of the web service providing the new items for the Model Info list are described in the (optional) “model_info” item of the extension file:

- URL (string; required): when the user opens the Model Info dialog, BIMx sends an HTTP(S) GET request to the URL provided, waits for the response (a JSON file, see specification below), parses the response and adds items to the Model Info popover based on the response. The URL might contain variable names; those will be substituted prior to sending the request. Variable values will be URL (percent) encoded. Please note that only HTTP (http:// and https://) URLs are supported for the time being. The HTTP request will be cancelled if the user closes the Model Info popover before the response is received. Currently only HTTP basic authentication (other than HTTP request header fields, see below) is supported, i.e. the request should either work without any further authentication challenge or require the presence of the “Authorization” HTTP request header field with “Basic <Base64 encoded username:password>” value. If the HTTP request fails for some reason (non-2xx status code, timeout, etc.), then a brief error message will be displayed for the user unless it’s authentication related (see below).
- HTTP header fields (array of strings, optional): in order to protect data provided by the web service and/or identify the app making the request, you could define custom HTTP header field values (e.g. “Authorization: key=<your_authorization_key>”)
- Authentication (string, optional): the only accepted value is “basic”, any other value will be ignored. See details below.

Example code snippet

```
…
"model_info": {
 "url": "https://www.yourwebservice.com/model?id=$(project_id)",
 "http_headers": [
   {
     "field": "Accept",
     "value": "application/json"
   },
   {
     "field": "Authorization",
     "value": "key=hG7k6ZG8bsd6gHK89a"
   }
 ],
 "authentication": "basic"
}
```

Response JSON format specification

The JSON file sent as the response by the web service should be encoded using UTF-8 encoding. It should contain an array of items. (If the response is empty, then no item will be added to the Model Info list.) For each item, you should provide the following details:

- Key (string, required): the title to be shown as the title (like “Project name” in the screenshot above); formatting (font, colour, alignment, etc.) will match the formatting of the items “built-in” to BIMx
- Value (string, required): the value to be shown below the title (like “Reconstruction of Villa Balassa” in the screenshot above); formatting (font, colour, alignment, etc.) will match the formatting of the items “built-in” to BIMx. Note that every key-value pair will be displayed in its own “section” (just like “Project name” and “Client full address” in the screenshot above)

Example response JSON


```
"items": [
 {
   "key": "Project FM ID",
   "value": "A-769876"
 },
 {
   "key": "Total maintenance hours",
   "value": "239.6 h"
 }
]
…
```


Set camera position with URL parameters

Using the appropriate bimxapplication:// URL you can instruct BIMx to open a given model, open its 3D model and set a certain camera position (including the direction the camera is looking at). This could be done with or without selecting elements. Below you can find the format of the URLs.

bimxapplication://showElementIn3dContext?hypermodel=<name_of_hyper_model>&

element_guids=<comma_separated_list_of_element_GUIDs>&

cameraPos=<camera_position>&

viewDirection=<direction_the_camera_is_looking_at>

Specification of parameters:

- hypermodel (string, required): the name of the hyper model to be opened
- element_guids (string, optional): a comma separated list of element GUIDs. The specified elements will be selected. If omitted, no element will be selected and the previous selection will remain unchanged.
- cameraPos: (string, optional): it specifies the x, y and z coordinates of the camera. The coordinates has to be separated with a comma (",") character (or it's percent-encoded equivalent "%2C") and has to use period (".") as the decimal separator. A sample value is "-6.34521007,11.1813793,13.561965". If viewDirection is not specified, but element is, then the camera will look at the centre of the specified element's bounding box. If multiple elements are specified (and viewDirection is missing), then the camera will look at the centre of the first element.
- viewDirection: (string, optional): it specifies the direction the camera is looking at (and not the coordinates of the model point the camera is looking at!). For example "0,0,1" means the camera will look up in the sky, regardless of the camera position set. In order to make sure camera animations work properly, please make sure the length of

the viewDirection vector is 1.0 (i.e. it's normalized). If cameraPos is not specified, then viewDirection will be ignored.

Option to download and open a Hyper-model from your own storage space

A new method is available to download and open a BIMx Hyper-model from any web-enabled public or private cloud storage. The prerequisite of this automated download is that the Hyper-model file has to be available on a basic http or https URL without any further query parameter. For example, this link downloads the *BIMx Demo Hyper-model* from the BIMx Model Transfer service:

https://bimx-storage.graphisoft.com/Download/0d020f3f-d95b-4941-8de1-0fd18b3bdf48

Not only to download but also to open the Hyper-model in BIMx app on mobile device you have to follow these rules to setup the OpenIn URL:

- The mandatory protocol is *bimxapplication://*
- The required query parameters are as follow:
  - *name*: - the name of the Hyper-mode (text, URL encoded)
  - *folderId*: - the unique identifier of the Hyper-model on your model storage server (integer)
  - *type*: HyperModel
  - *protocol*: - optional parameter (text, default: “https”)
  - *trsite*: 1

According to these rules the proper OpenIn URL for the the *BIMx Demo Hyper-model* is:

bimxapplication://bimx-storage.graphisoft.com/Download/ 0d020f3f-d95b-4941-8de1-0fd18b3bdf48?

name= BIMx%20Demo%20Hyper-model&

folderId=122826&

type=HyperModel&

trsite=1&

protocol=https

Use a hyperlink to open a Hyper-model

You can define a URL which will open a BIMx Hyper-model without element selection. The template for hyperlinks looks like this:

bimxapplication://showHypermodelContents?hypermodel=hypermodel_name

The first part contains GRAPHISOFT’s multi-platform ‘custom url scheme’ technique. This scheme is registered by the OS when you install the BIMx app on your mobile device. From that point on the OS will recognise that this URL is associated with BIMx, therefore it will Open the BIMx app when the link is tapped.

The second part defines which BIMx Hyper-model to open from your available Hyper-models. The hypermodel_name part needs to be replaced with the name you defined for the Hyper-

model. If its name contains any special characters it needs to be converted with the default URL encoding (https://en.wikipedia.org/wiki/Percent-encoding).

Use a hyperlink to reach a specific model element in the 3D model

An element access hyperlink consists of three individual parts. The template for hyperlinks looks like this:

bimxapplication://showElementIn3dContext?

hypermodel=hypermodel_name&

element_guids=Unique_ID1,Unique_ID2,…,Unique_IDn

The first part contains GRAPHISOFT's multi-platform 'custom url scheme' technique. This scheme is registered by the OS when you install the BIMx app on your mobile device. From that point on the OS will recognize that this URL is associated with BIMx, therefore it will Open the BIMx app when the link is tapped.

The second part defines which BIMx Hyper-model to open from your available Hyper-models. The hypermodel_name part needs to be replaced with the name you defined for the Hyper-model. If its name contains any special characters it needs to be converted with the default URL encoding (https://en.wikipedia.org/wiki/Percent-encoding)

The third part contains the Unique_ID of the element that you want to access with the hyperlink. This ID can be obtained in ARCHICAD, by listing the Unique ID field in the Interactive Schedule. The built-up hyperlink can contain multiple Unique_IDs, so BIMx will open the 3D model with all listed elements, highlighted.

Set element group highlights with multiple colors

Using the "showElementIn3dContext" command of the bimxapplication:// custom URL schema you can use additional colors (besides the default green selection color) to highlight certain elements in the 3D model. Currently these additional colors are only available via the "showElementIn3dContext" command.

You can define up to 8 groups of elements, each group having its own color. You can optionally define a name for each selection group. The name of the selection group will be shown in the selection menu.

For each group you have to provide a comma-separated list of element GUIDs. The name of the URL parameters for those element GUID lists are group1, group2, group3, ... group8. You can use whichever group you like, there is no need to start with group1.

To define the name of a selection group, please use groupName1, groupName2, ... groupName8 URL parameters.

Each group has a hardwired color as follows:

- Group 1: red
- Group 2: blue
- Group 3: orange
- Group 4: cyan
- Group 5: teal
- Group 6: pink
- Group 7: yellow
- Group 8: purple

Please note that you always must define the name of the hyper model in the URL, otherwise it won't work.

Example hyperlink:

bimxapplication://showElementIn3dContext?

hypermodel=BIMx%20Demo%20Hyper-model&

group1=d3310ef3-dabf-4bd2-993e-5b325be2700e,809941fb-5d8d-4da6-8e3c-665f190fef04&

groupName1=Red+elements&

group2=fc2d314d-3601-4789-908b-26bbec17a473&

groupName2=Blue+elements

Option to define an external link to a 2D document

Using the "show2dDocument" command of the bimxapplication:// custom URL schema you can have BIMx open a layout. In the URL you have to specify the name of the hyper model (hypermodel URL parameter) and the GUID of the layout (layoutGuid URL parameter) you would like to open.

Note: To get access to the Layout GUIDs, please use your ArchiCAD Add-On to obtain these data on ARCHICAD API.

Example hyperlink:

bimxapplication://show2dDocument?

hypermodel=BIMx%20Demo%20Hyper-model&

layoutGuid=84f52ab0-a7d8-ef78-b35c-b1e73f579f38

Web forms demo

We have created a simple web application and a BIMx extension to demonstrate how you could “link” a web application and BIMx. Using this demo application, you can save/view remarks to/of Hyper-models and 3D model elements.

To give it a try you have to install the demo extension first. To do so, please visit the

http://bimx-ext-test.graphisoft.com/

website using your mobile device (which has BIMx installed) and tap on the “Install the corresponding demo extension” link. Once the extension has been successfully installed (you should see a confirmation message in BIMx) you can either open a demo web form for the model (open a hyper model > open the “...” menu > tap on “Demo - remark”) or another demo web form for a selected element (open a hyper-model > open 3D > select an element > open the element context menu > tap on “Demo - remark”). In the web form you can save a textual remark for the model or for the selected element.

Please note the forms in the demo web application use a shared database, i.e. any change you make (e.g. changing the remark for the demo hyper model) will be visible to all users on all platforms.

Please also note that the demo web application is running on a low-priority web server meaning that it might take a few seconds for the web application to spin up if it isn’t used for an extended period of time.