Introduction to The BIM 360™ Glue® API

Mikako Harada
AEC Workgroup Technical Lead & Americas Manager
Autodesk Developer Network

Manu Venugopal
Integration Engineer
Autodesk Consulting
Class summary

In this class, we introduce you to the BIM 360 Glue API. We will talk about what Glue Web Services API is and show you how to use them. We will show you how to embed Glue display components. We demonstrate sample applications, and instruct you step by step how to get started with programming using Glue API.
Key learning objectives

At the end of this class, you will be able to:

- Understand the basic structure of the BIM 360 Glue software REST API
- Learn how to use the BIM 360 Glue software web services API to access your BIM 360 Glue software data
- Learn how to use the display components API to embed the BIM 360 Glue software view in your application
- Understand current capabilities and limitations of the BIM 360 Glue software API
Prerequisite

This class assumes you have a basic understanding of web programming, and familiar with terminologies related to web services.

Working knowledge of the REST API is helpful for understanding the discussion.

Actual web programming experience may not be necessary if you are interested in getting started with Glue API programming.

You will hear a lot of buzzwords if you are not familiar with programming.
Classes about BIM 360 Glue

If you are interested in more on products usage, business, and use cases...

- CO5007-R – “Return on Investment of Cloud-Based Collaboration and Mobility for Construction.” Thurs, Dec 4 8-9:30AM. Round Table

- CO6084-L – “BIM 360 Glue: The Super Glue for Your Model Coordination Workflows.” Thurs, Dec 4, 10-11:30AM. Hands-on Lab

- CO5528 – “Building Information Modeling for Construction Safety.” Tues, Dec 2, 5-6PM

Agenda

- BIM 360 Glue API Overview
  - Web Services
  - Display Component
- Where to Begin
  - Resources
  - “My First Glue Application” – Step-by-step instructions
- Summary
- Q&A
BIM 360 Glue API Overview
BIM 360 Glue API

- **Web Services API**
  - Glue’s REST API provides a RESTful interface that returns JSON or XML
  - A data access API allowing developers to query and modify various data objects within the Glue platform
  - Developers can integrate external applications such as project management systems, accounting systems or custom developed solutions

- **Display Component / Viewer**
  - The Glue Viewer is an embeddable component used to show 3D models from the Glue platform
  - Allows for developers to customize the viewer experience
Web Services API: Service Groups

- Security Service
- Project Service
- Model Service
- User Service
- Action Service
- System Service
- Points Services
- Analytics Services
The Glue Platform stores information about many user interactions within the system. These operations are referred to as “Actions.” Typical Actions would be:

- Uploading a Model
- Creating a View on a Model
- Adding Mark Ups to a Model
- Creating a Clash report for a Model

The Glue Web Services API returns many of these actions in API responses. Actions can be loaded to the viewer to show the user the exact view/state/etc… of a model when the creator of the Action performed the operation.
The Glue Display Component allows users to place 3D Models from the Glue Platform on a webpage or embedded browser.

- This is done using an `<iframe>` element
- The following is a sample code snippet for a Viewer embed:

```
<iframe title="Glue Viewer" width="800" height="600" src="glue_viewer_url..." frameborder="0" allowfullscreen></iframe>
```
Where to Begin
Glue Developer Site

- Glue Developer Site: [www.autodesk.com/developglue](http://www.autodesk.com/developglue)
- Hope to add more materials as they become available
**References** (Links on Glue Developer Page)

- Glue API Developer Page
  [www.autodesk.com/developglue](http://www.autodesk.com/developglue)
- API documentation main page
- Viewer: List of Supported File Formats
- Add-ins for AutoCAD (including Verticals), Revit and Navisworks
- Product Help
- Product Page
- Viewer documentation and the viewer base URL (* need up to date ⚠)
- Samples
Additional References

- “The 360 View” Blog by Manu Venugopal
  - http://the360view.typepad.com/blog/
- AEC DevBlog
  - http://adndevblog.typepad.com/aec/
  - e.g., migrated samples posted here: http://adndevblog.typepad.com/aec/2013/10/bim-360-glue-api-pilot-and-updated-samples.html

- Note: Samples on SDK site may not be up to date
- Hope the API community grows, and get resource and gain momentum …
To use Glue API, you will need:

- BIM 360 Glue account
- **API key and secret**
- Programming environment of your choice (e.g., MSVS for C#, Editor for JavaScript)

At a moment, not everybody can have access to dev env

- Already Glue customer → Obtain through the contract
- ADN member → ADN staging access through ADN. Request through DevHelp Online
- Note: Free trial (https://b4.autodesk.com/freetrial/) does not include API access
- What about today’s AU participants?
SDK Samples

- Documentations (and download)
- Latest samples download
  http://adndevblog.typepad.com/files/glue-api-samples.zip

- Two samples:
  - Test Harness – Windows desktop client sample in C#.
  - Sample Web Application – ASP.NET. Web application in C#
Windows Desktop Client (Test Harness)
ASP .NET Web Application
“My First Glue Application”
Glue API Intro Labs Exercises

Incremental code project to learn the basics of Glue API

Lab1 – “Hello Glue World” sign-in REST call
Lab2 – retrieve a model id and display in embedded viewer
Lab3 – web application version
Lab4 – JavaScript layer for selection and property access

All minimum for clarity for learning purposes
In C# and JavaScript
Lab 1 - “Hello Glue World”

- Minimum sample to learn the basics of REST API call
- Simple Windows Form App in C#
- RestSharp library for simplicity and clarity
- Sign-in Glue REST call
- Services used:
  - Security/login
Lab2 – Glue API Intro

- Retrieve a model and display in an embedded viewer
- Simple Windows Form App in C#
- RestSharp
- Services used:
  - Security/login
  - Project list
  - Model list
  - Display component (or viewer)
Lab 3 – Glue API Web Intro

- Simple ASP.NET Web application
- Retrieve a model and display in an embedded viewer
- RestSharp
- Services used:
  - Security/login
  - Project list
  - Model list
  - Display component (or viewer)
Lab 4 – Glue API Web Intro JS

- JavaScript layer
- Selection and property access
- Simple ASP.NET Web application
- Event handlers and post messages between web page and viewer
  - $(document).on('selectionchanges', f(e))
  - $(document).on('gotproperties', f(e))
  - GlueEmbedded.zoomSelection()
  - GlueEmbedded.getSelectedProperties()
Lab1 – “Hello Glue World”
Lab1 - “Hello Glue World”

- Minimum sample to learn the basics of REST API call
- Simple Windows Form App in C#
- RestSharp library for simplicity and clarity
- Sign-in Glue REST call
- Services used:
  - Security/login
1. Login - Request

- Authenticates a user to the Glue platform and returns an authentication token (**auth_token**) for subsequent service calls.
- Supported Request Methods: Post
- Required Parameters
  - format
  - login_name
  - password
  - company_id
  - api_key
  - timestamp
  - signature (sig)
1. Login - Response

Two types of format for response:

- **JSON**
  ```json
  {
    "auth_token":"76d47d99102b42e3abb30851e63d1a67",
    "user_id":"36477f1a-3366-44f5-bf0a-3a4a019c52bd"
  }
  ```

- **XML**
  ```xml
  <?xml version="1.0" encoding="utf-8"?>
  <security_login_response_v1 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <auth_token>76d47d99102b42e3abb30851e63d1a67</auth_token>
    <user_id>36477f1a-3366-44f5-bf0a-3a4a019c52bd</user_id>
  </security_login_response_v1>
  ```

Note: Line break added for readability
public string Login(string login_name, string password)
{
    // Calculate signature components.
    string timeStamp = Util.GetUNIXEpochTimestamp().ToString();
    string signature = Util.ComputeMD5Hash(apiKey + apiSecret + timeStamp);

    // (1) Build request
    var client = new RestClient();
    client.BaseUrl = baseApiUrl;

    // Set resource/end point
    var request = new RestRequest();
    request.Resource = "security/v1/login.json";
    request.Method = Method.POST;

    // Set required parameters
    request.AddParameter("login_name", login_name);
    request.AddParameter("password", password);
    request.AddParameter("company_id", companyId);
    request.AddParameter("api_key", apiKey);
    request.AddParameter("timestamp", timeStamp);
    request.AddParameter("sig", signature);

    // (2) Execute request and get response
    IRestResponse response = client.Execute(request);

    // Get the auth token.
    string authToken = "Undefined";
    if (response.StatusCode == HttpStatusCode.OK)
    {
        JsonSerializer deserial = new JsonSerializer();
        LoginResponse loginResponse = deserial.Deserialize<LoginResponse>(response);
        authToken = loginResponse.auth_token;
    }
    return authToken;
}
To define parameters:

- `timestamp = Utils.GetUNIXEpochTimestamp()` returns the number of seconds since January 1st, 1970 00:00:00 GMT.

- `signature = Utils.ComputeMD5Hash(apiKey + apiSecret + timeStamp)`
  
REST Client Calls

- To make a REST call
  - RestSharp utility library. Simple REST and HTTP client API for .NET (http://restsharp.org/)
  - Create a REST client with a given url.
    ```csharp
    var client = new RestClient()
    client.BaseUrl = baseApiUrl
    ```
  - Set resource/end point, parameters
    ```csharp
    var request = new RestRequest()
    request.Resource = "security/v1/login.json"
    request.Method = Method.POST
    request.AddParameter("timestamp", timeStamp)
    ```
  - Get response from the web service
    ```csharp
    IRestResponse response = client.Execute(request);
    response.StatusCode
    response.ResourceUri.AbsoluteUri
    response.Content
    ```
To parse from response

```csharp
JsonDeserializer deserial = new JsonDeserializer();
LoginResponse loginResponse = deserial.Deserialize<LoginResponse>(response);

authToken = loginResponse.auth_token;
```

```csharp
[Serializable]
public class LoginResponse
{
    public string auth_token { get; set; }
    public string user_id { get; set; }
}
```

You may use other utilities that you may want to take advantage of
Lab2 – Glue API Intro
Lab2 – Glue API Intro

- Retrieve a model and display in an embedded viewer
- Simple Windows Form App in C#
- RestSharp
- Services used:
  - Security/login
  - Project list
  - Model list
  - Display component (or viewer)
Work Flow

1. Login >> auth_token

2. Get a list of projects >> project_id

3. Get a list of model from a given project >> model_id

4. Display a model
2. Project List - Request

- Returns a list of project from the Glue platform.
- URL: [https://b4.autodesk.com/api/project/v1/list.{format}](https://b4.autodesk.com/api/project/v1/list.{format})
- Supported Request Methods: GET
- Required Parameters
  - format
  - company_id
  - api_key
  - timestamp
  - signature (sig)
  - auth_token (obtained from login)

- Doc: [https://b4.autodesk.com/api/project/v1/list/doc](https://b4.autodesk.com/api/project/v1/list/doc)
2. Project List – Response

{
  "project_list": [
    {
      "folder_tree":null,
      "project_roster":null,
      "project_id": "The BIM 360 Glue Project ID",
      "project_name": "The name for the project (URL Encoded)",
      "company_id": "The Company ID for the Project",
      "created_date": "The date the Project was added",
      "modify_date": "The date the Project name was last modified",
      "start_date": "The date the Project was started",
      "end_date": "The date the Project ends",
      "last_activity_date": "The last date there was Project activity"
    },
    {
      "folder_tree":null,
      "project_roster":null,
      "project_id": "The BIM 360 Glue Project ID",
      ...
    }
  ],
  "page":1,
  "page_size":2,
  "total_result_size":2,
  "more_pages":0
}
public List<Project> ProjectList(string authToken)
{
    string timeStamp = Utils.GetUNIXEpochTimestamp().ToString();
    string signature = Utils.ComputeMD5Hash(apiKey + apiSecret + timeStamp);

    // (1) Build request
    var client = new RestClient();
    client.BaseUrl = baseApiUrl;

    // Set resource or end point
    var request = new RestRequest();
    request.Resource = "project/v1/list.json";
    request.Method = Method.GET;

    // Add parameters
    request.AddParameter("company_id", companyId);
    request.AddParameter("api_key", apiKey);
    request.AddParameter("timestamp", timeStamp);
    request.AddParameter("sig", signature);
    request.AddParameter("auth_token", authToken);

    // (2) Execute request and get response
    IRestResponse response = client.Execute(request);

    if (response.StatusCode != HttpStatusCode.OK)
    {
        return null;
    }

    // Get a list of projects.
    JsonDeserializer deserial = new JsonDeserializer();
    ProjectListResponse projListResponse = deserial.Deserialize<ProjectListResponse>(response);
    List<Project> proj_list = projListResponse.project_list;

    return proj_list;
}
3. Model List – Request

- Returns a list of models from the Glue platform.
- URL:  [https://b4.autodesk.com/api/model/v1/list.{format}](https://b4.autodesk.com/api/model/v1/list.{format})
- Supported Request Methods: GET
- Required Parameters
  - format
  - company_id
  - api_key
  - timestamp
  - signature (sig)
  - `auth_token` (obtained from login)
  - `project_id` (obtained from the previous call)
- Doc: [https://b4.autodesk.com/api/project/v1/list/doc](https://b4.autodesk.com/api/project/v1/list/doc)

Glue Web Service API requests includes these 5 parameters: format, company_id, api_key, timestamp, signature (sig). Additional parameter is `project_id` (obtained from the previous call).
3. Model List – Response

```json
{
  "model_list": [
    {
      "company_id": "The company identifier for this model",
      "project_id": "The Project identifier for this model",
      "model_id": "The model identifier for the model",
      "model_version": 1,
      "model_version_id": "The version identifier for this specific version of the model",
      "model_name": "The name for the model",
      "created_by": "The login_name of the creator of the model",
      "created_date": "The date of model was created",
      "modified_by": "The login_name of the last user to modify the model",
      "modified_date": "Date of last modification",
      "published": 0
    },
    {
      "company_id": "The company identifier for this model",
      ...
    }
  ],
  "page": 1,
  "page_size": 2,
  "total_result_size": 2,
  "more_pages": 0
}
```
public List<ModelInfo> ModelList(string authToken, string projectId)
{
    string timeStamp = Utils.GetUNIXEpochTimestamp().ToString();
    string signature = Utils.ComputeMD5Hash(apiKey + apiSecret + timeStamp);

    // (1) Build request
    var client = new RestClient();
    client.BaseUrl = baseApiUrl;

    // Set resource or endpoint
    var request = new RestRequest();
    request.Resource = "model/v1/list.json";
    request.Method = Method.GET;

    // Add parameters
    request.AddParameter("company_id", companyId);
    request.AddParameter("api_key", apiKey);
    request.AddParameter("timestamp", timeStamp);
    request.AddParameter("sig", signature);
    request.AddParameter("auth_token", authToken);

    // (2) Execute request and get response
    IRestResponse response = client.Execute(request);
    if (response.StatusCode != HttpStatusCode.OK)
    {
        return null;
    }

    // Get a list of models.
    JsonSerializer deserial = new JsonSerializer();
    ModellistResponse modellistResponse = deserial.Deserialize<ModellistResponse>(response);
    List<ModelInfo> model_list = modellistResponse.model_list;
    return model_list;
}
4. Viewer

- Allow users to place Glue 3D models on a web page.
- URL:  https://b2.autodesk.com?
- Required Parameters: 5 usual required parameters, plus two ways:
  1. "&runner=embedded/#" + company_id + "/" + project_id + "/" + model_id
  2. "&runner=embedded/" + company_id + "/action" + "/" + action_id
- 🕵️‍♂️ Note: doc currently not up to date. 😞
4. Viewer
4. Viewer Sample Code

```csharp
public string View(string authToken, string projectId, string modelId)
{
    string timeStamp = Utils.GetUNIXEpochTimestamp().ToString();
    string signature = Utils.ComputeMD5Hash(apiKey + apiSecret + timeStamp);

    string callArgs = "";
    // We need these 5 arguments for every subsequent requests.
    // Auth token is returned when you login.
    callArgs += "&company_id=" + HttpUtility.UrlEncode(companyId);
    callArgs += "&api_key=" + HttpUtility.UrlEncode(apiKey);
    callArgs += "&timestamp=" + HttpUtility.UrlEncode(timeStamp);
    callArgs += "&sig=" + HttpUtility.UrlEncode(signature);
    callArgs += "&auth_token=" + HttpUtility.UrlEncode(authToken);
    //
    callArgs += "&runner=embedded/#" + HttpUtility.UrlEncode(companyId)
                + "" + projectId + "" + modelId;

    // URL that we are going to embed a web browser.
    string url = baseViewerUrl + callArgs;
    return url;
}
```

```csharp
private void buttonView_Click(object sender, EventArgs e)
{
    string url = glueCall.View(m_authToken, m_project_id, m_model_id);
    // a view embedded form's web browser by URL.
    webBrowser1.Url = new System.Uri(url);
}
```
Lab3 – Glue API Web Intro
Lab3 – Glue API Web Intro

- Simple ASP.NET Web application
- Retrieve a model and display in an embedded viewer
- RestSharp
- Services used:
  - Security/login
  - Project list
  - Model list
  - Display component (or viewer)

More or less the same as Lab2
Web Application

- Glue Web Services REST calls – Reuse
- UI: WinFrom ➔ WebForms (HTML based UI)
- WebForm1.aspx

```csharp
protected void ButtonView_Click(object sender, EventArgs e)
{
    string authToken = HttpContext.Current.Session["authToken"] as string;
    string project_id = HttpContext.Current.Session["projectId"] as string;
    string model_id = HttpContext.Current.Session["modelId"] as string;

    string url = Glue.View(authToken, project_id, model_id);

    // embed a viewer in iframe
    iframeGlueSrc = url;
}
```
Lab4 – Glue API Web Intro JS
Lab4 – Glue API Web Intro JS

- JavaScript layer
- Selection and property access
- Simple ASP.NET Web application (same)
- Event handlers and post messages to interface between web page and viewer
  - $(document).on('selectionchanges', f(e))
  - $(document).on('gotproperties', f(e))
  - GlueEmbedded.zoomSelection()
  - GlueEmbedded.getSelectedProperties()
JavaScript Layer

- Thin JavaScript layer to interface with Glue viewer.
- jQuery – popular JavaScript library. Simplify query documents parts
- **GlueEmbedded** - Post message to the viewer:
  - Init(iframe)
  - setSelection(objectPath)
  - getProperties(objectPath)
  - getSelectedProperties()
  - zoomSelection()

- “Message” event from the viewer
  - $(document).on( *message*, function(e) {})
    - ‘selectionchanged’
    - ‘gotproperties’
$(document).ready(function () {
    // initialize the embedded viewer passing in the target iframe window
    GlueEmbedded.init(window.frames[0]);

    // Setting event handlers to get messages from the viewer.
    // specify an event handler on the document for the "selectionchanged" from the viewer.
    // The event's eventData property will be set to the array of selected objects
    $(document).on('selectionchanged', function (e) {
        $("#messages").val("object selected: " + JSON.stringify(e.eventData));
    });

    // specify an event handler on the document for the "gotproperties" event from the viewer.
    // The event's eventData property will be set to the collection of object properties
    $(document).on('gotproperties', function (e) {
        $("#messages").val("got properties: " + JSON.stringify(e.eventData));
    });

    // Posting a message to the viewer.
    // When the user click get_properties button, this is called.
    $('#get_properties').on('click', function () {
        GlueEmbedded.getSelectedProperties();
    });

    // This is for zooming in a currently selected element.
    $('#zoom_selection').on('click', function () {
        GlueEmbedded.zoomSelection();
    });
});
Additional Sample

- GlueSDKSampleWebApp
Conclusions
Learning Objectives

You have learned how to:

- Understand the basic structure of the BIM 360 Glue software REST API
- Learn how to use the BIM 360 Glue software web services API to access and modify your BIM 360 Glue software data
- Learn how to use the display components API to embed the BIM 360 Glue software view in your application
- Understand current capabilities and limitations of the BIM 360 Glue software API
Q. What do I need to do to try Glue API?

- Already a Glue customer → request through the contract
- ADN member → Staging request through DevHelp Online
- Today’s AU Class participants
  - Send us an e-mail
  - Will send you keys and an instruction