Пошаговое Руководство

Contents

Шаг 1: Регистрация и создание приложения

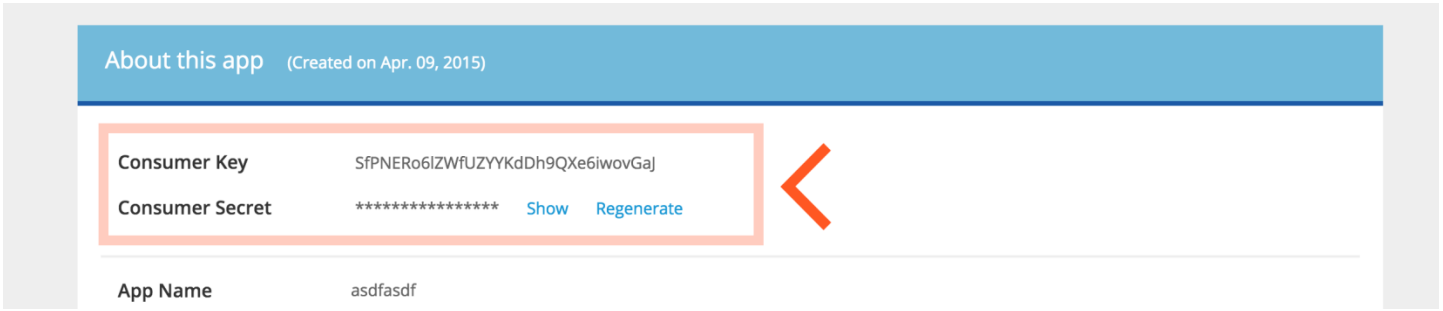
Чтобы начать работу с Autodesk View and Data API, необходимо создать аккаунт Autodesk. Для этого нажмите "Sign Up" на панели навигации и заполните необходимую информацию.

После регистрации вы можете перейти к созданию приложения.

- Нажмите кнопку "Create an App" на панели навигации или на странице "My Apps".
- Укажите название приложения, его описание, а также подставьте URL обратного вызова (callback URL).
- Выберите API – "View and Data API".

Примечание:
URL обратного вызова нужен для двухэтапной аутентификации по протоколу OAuth2. Ничего страшного, если обратный вызов (callback) у вас пока не настроен, однако для продолжения процедуры это поле должно быть заполнено.

На странице только что созданного приложения вы увидите Consumer Key и Consumer Secret – ключ и секрет, необходимые для авторизации вашего приложения. Они будут отправлены на электронный адрес, указанный в вами при регистрации аккаунта. Сохраните эти данные в надежном месте – они вам понадобятся на следующем шаге.



Шаг 2: Получение токена доступа

В этом первом тесте мы будем создавать все запросы, используя cURL (http://curl.haxx.se/). В случае клиентских приложений на других языках все принципы и синтаксис запросов останутся теми же.

Для получения токена доступа через Authentication API введите ваши Consumer Key (в качестве client_id) и Consumer Secret (в качестве client_secret).

```
curl --data "client_id=obQDn8P0GanGFQha4ngKKVWcxwyvFAGE&client_secret=eUruM8HRyc7BAQ1e&grant_type=client_credentials"
\
https://developer.api.autodesk.com/authentication/v1/authenticate --header "Content-Type: application/x-www-form-
urlencoded" -k
```

Описание параметров запроса приведены здесь Create an OAuth2 token. Ответ сервера должен иметь статус 200 со строкой json следующего вида:

```
{
  "token_type" : "Bearer",
  "expires_in" : 899,
  "access_token" : "GX6OONHlQ9qoVaCSmBqJvqPFUT5i"
}
```

Ответ содержит токен доступа, его тип и время жизни (в секундах). Ваше приложение должно сохранить эту информацию и обеспечить получение нового токена доступа, как только истечет срок действия прежнего.

Шаг 3: Создание «ведра»

"Ведра" (buckets) – это контейнеры для данных, используемые Autodesk View and Data API. До загрузки файлов создайте "ведро" и установите срок хранения данных. Вам доступны следующие варианты :

- **Transient** – кратковременное хранилище, которое существует всего 24 часа и идеально подходит для быстрых тестов.
- **Temporary** – временное хранилище, которое существует 30 дней. Подходит для данных, которые требуют загрузки и доступа лишь в течение ограниченного времени. Этот тип размещения данных позволяет несколько сэкономить затраты.
- **Persistent** — постоянное хранилище, которое существует, пока вы сами его не удалите. Объекты, обращение к которым не осуществлялось в течение 2 лет, могут быть заархивированы.

Существуют некоторые ограничения на множество символов, которые можно использовать в имени контейнера (ведра). К примеру, буквы допустимы лишь в нижнем регистре. Подробности приведены в документации по API.

Создайте "ведро" с помощью метода POST /oss/{version}/buckets API
(см. раздел *OSS Bucket and Object API v2.0*):

```
curl -k --header "Content-Type: application/json" --header "Authorization: Bearer GX6OONHlQ9qoVaCSmBqJvqPFUT5i" \
--data "{\"bucketKey\":\"mybucket\",\"policy\":\"transient\"}" https://developer.api.autodesk.com/oss/v2/buckets
```

Примечание: В Windows приходится использовать escape-последовательность “\” для кодирования внутренних двойных кавычек, как показано в примере выше. В других операционных системах можно записать содержательную часть строки json как: ‘{"bucketKey":"mybucket","policy": "transient"}’ .

В этом примере параметр заголовка "Authorization: Bearer" содержит токен доступа, полученный нами на Шаге 2.

После создания контейнера проверьте, что он действительно существует, с помощью Get Bucket Details API:

```
curl -k -H "Authorization: Bearer GX6OONHlQ9qoVaCSmBqJvqPFUT5i" -X GET \
https://developer.api.autodesk.com/oss/v2/buckets/mybucket/details
```

В случае успеха ответ сервера выглядит так:

```
{
    "key" : "mybucket",
    "owner" : "obQDn8P0GanGFQha4ngKKVWcxwyvFAGE",
    "createDate" : 1401735235495,
    "permissions" : [{
                "serviceId" : "obQDn8P0GanGFQha4ngKKVWcxwyvFAGE",
                "access" :  "full"
          }
    ],
    "policyKey" : "transient"
}
```

Шаг 4: Загрузка файла

Следующий шаг – выгрузка файла в ваш контейнер для подготовки к последующему просмотру. Непосредственно после загрузки файл регистрируется в службе просмотра. В настоящее время на входе поддерживаются более 60 файловых форматов, включая:

- Autodesk DWG™
- Autodesk Inventor®
- Fusion 360™
- SIM 360™
- Autodesk Navisworks®
- Autodesk Revit®
- Autodesk 3Ds Max®
- Solidworks®
- CATIA®
- Siemens Parasolid™
- Siemens NX™
- Siemens OpenJT™
- WaveFront Technologies OBJ

Получить актуальный список поддерживаемых форматов можно с помощью **GET /viewingservice/{version}/supported** API (*Supported API*)

curl -k --header "Authorization: Bearer lEaixuJ5wXby7Trk6Tb77g6Mi8IL" https://developer.api.autodesk.com/viewingservice/v1/supported

Команда возвращает массив расширений, для которых возможна трансляция:

```
"extensions" : ["ipt", "neu", "stla", "stl", "xlsx", "jt", "jpg", "skp", "prt", "dwf", "xls",
      "png", "sldasm","step", "dwg", "zip", "nwc", "model", "sim", "stp", "ste", "f3d",
      "pdf", "iges", "dwt", "catproduct", "csv", "igs", "sldprt", "cgr", "lll", "3dm", "sab",
      "obj", "pptx", "cam360", "jpeg", "bmp", "exp", "ppt", "doc", "wire", "ige", "rcp",
      "txt", "dae", "x_b", "3ds", "rtf", "rvt", "g", "sim360", "iam", "asm", "dlv3", "x_t",
      "pps", "session", "xas", "xpr", "docx", "catpart", "stlb", "tiff", "nwd", "sat", "fbx",
      "smb", "smt", "dwfx", "tif"],
      ...
```

Расширения с префиксом "Viewing-*" поддерживаются для просмотра.

Файл загружается на сервер с помощью запроса **PUT /oss/{version}/buckets/{bucketname}/objects/{filename}** API (*OSS Upload API v2.0*):

```
curl --header "Authorization: Bearer K16B98iaYNElzVheldlUAUqOoMRC" --header "Content-Length: 308331" \
-H "Content-Type:application/octet-stream" --header "Expect:" \
--upload-file "skyscpr1.3ds" -X PUT https://developer.api.autodesk.com/oss/v2/buckets/mybucket/objects/skyscpr1.3ds -k
```

Важно корректно задать заголовки `Content-Length` и `Content-Type`, а также оставить пустым заголовок `Expect`. Последний используется данным API в случае загрузки больших файлов. Более подробную информацию вы можете найти в документации.

В случае успешной загрузки файла ответ сервера будет выглядеть примерно так:

```
{
  "bucket-key": "mybucket",
  "objects": [
```

```
    {
        "location": "https://developer.api.autodesk.com/oss/v2/buckets/mybucket/objects/skyscpr1.3ds",
        "size": 308331,
        "key": "skyscpr1.3ds",
        "id": "urn:adsk.objects:os.object:mybucket/skyscpr1.3ds",
        "sha-1": "e84021849a9f5d1842bf792bbcbc6445c280e15b",
        "content-type": "application/octet-stream"
    }
  ]
}
```


Для нас в этом ответе самым важным элементом является **id**. Он понадобится на следующих этапах.

Если ваша модель состоит из нескольких файлов (например, файл сборки Inventor IAM и несколько файлов деталей IPT этой сборки), загрузите в ваш контейнер каждый из этих файлов и сохраните их id.

Шаг 5: Задание связей между различными файлами

Этот шаг необходим, только если ваша модель состоит из набора различных файлов. Перед тем, как зарегистрировать файлы в сервисе Data and Viewing, вы должны указать связи (отношения) между ними. Для это цели используется References API.

Этот API использует формат json для задания URN главного документа и URN одного или нескольких связанных с ним документов. POST /references/{version}/setreference API (*References Service*).

Создайте файл json с информацией о связях внутри вашей модели как показано в следующем примере:

```
{
  "master" : "urn:adsk.objects:os.object:alexbicalhobucket2/A1.iam",
  "dependencies" : [
    { "file" : "urn:adsk.objects:os.object:alexbicalhobucket2/A1A1.iam",
      "metadata" : {
          "childPath"  : "A1A1.iam",
          "parentPath" : "A1.iam"
      }
    },
    { "file" : "urn:adsk.objects:os.object:alexbicalhobucket2/A1P1.ipt",
      "metadata" : {
          "childPath"  : "A1P1.ipt",
          "parentPath" : "A1.iam"
      }
    },
    { "file" : "urn:adsk.objects:os.object:alexbicalhobucket2/A1P2.ipt",
      "metadata" : {
          "childPath"  : "A1P2.ipt",
          "parentPath" : "A1.iam"
      }
    }
    ]
}
```

Используйте URN главного (master) и связанных с ним файлов, полученные вами на Шаге 4. Дочерне/родительские отношения (child/parent relationships) следует задать в параметрах "childPath" и "parentPath". Если требуется, укажите дополнительные подчинения. В этом примере файлы деталей A1P1.ipt and A1P2.ipt, а также объединяющая их подсборка A1A1.iam, подчинены сборке верхнего уровня A1A.iam, которая в данном случае является главным файлом.

```
curl -k -H "Content-Type: application/json" -H "Authorization:Bearer GX6OONHlQ9qoVaCSmBqJvqPFUT5i" \
-i -d @references.json https://developer.api.autodesk.com/references/v1/setreference
```

Шаг 6: Регистрация данных во Viewing Services

Как только файл загружен на сервер, его необходимо зарегистрировать в службе просмотра. Регистрация запускает процесс трансляции данных для подготовки их к просмотру: POST /viewingservice/{version}/register API (Registration API):

```
curl -k -H "Content-Type: application/json" -H "Authorization:Bearer GX6OONHlQ9qoVaCSmBqJvqPFUT5i" \
-i -d "{\"urn\":\"dXJuOmFkc2sub2JqZWN0czpvcy5vYmplY3Q6bXlidWNrZXQvc2t5c2NwcjEuM2Rz\"}" \
https://developer.api.autodesk.com/viewingservice/v1/register
```

Этот API в теле json принимает единственный параметр – URN загруженного на сервер файла. Однако здесь URN должен быть представлен в кодировке **Base64**. Именно такое представление URN будет использоваться нами на всех последующих этапах. Для перекодирования URN в формат Base64 вы можете воспользоваться каким-нибудь online-приложением (напр., http://www.base64encode.org/).

Примечание:

Убедитесь, чтобы символы перевода на новую строку CR/LF (Carriage Return / New Line) не попали на вход утилите перекодирования. API не предполагает использования символов новой строки.

После регистрации файла можно проверить состояние процесса трансляции запросом GET /viewingservice/{version}/{urn}/status (*Get Viewable*) для указанного URN:

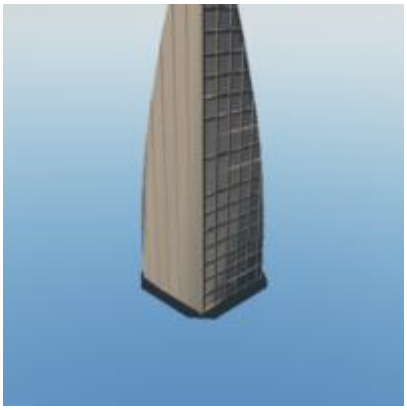
```
curl -k -i -H "Authorization: Bearer GX6OONHlQ9qoVaCSmBqJvqPFUT5i" -X GET \
https://developer.api.autodesk.com/viewingservice/v1/dXJuOmFkc2sub2JqZWN0czpvcy5vYmplY3Q6bXlidWNrZXQvc2t5c2NwcjEuM2Rz/
status
```

Ответом будет json с информацией как о документе в целом, так и о его компонентах. Обратите внимание, что даже если лишь некоторые компоненты имеют статус "complete", зарегистрированные данные уже доступны к просмотру.

По окончании трансляции можно получить изображение предпросмотра (thumbnail): GET /viewingservice/{version}/thumbnails/{urn} API (*Get Thumbnails*):

```
curl -k -H "Authorization: Bearer GX6OONHlQ9qoVaCSmBqJvqPFUT5i" -X GET \
https://developer.api.autodesk.com/viewingservice/v1/thumbnails/dXJuOmFkc2sub2JqZWN0czpvcy5vYmplY3Q6bXlidWNrZXQvc2t5c2
NwcjEuM2Rz  > C:\Users\Public\thumb.png
```

В моем случае это изображение выглядит так:



Шаг 7: Загрузка URN в JavaScript viewer

До сих пор мы использовали различные Autodesk View and Data APIs, применяя в примерах curl. Теперь же мы займемся JavaScript API для настройки просмотра непосредственно в браузере.

Примечание:
Существует необязательный (optional) параметр, который определяет, какую именно версию файла предполагается просматривать. Это может пригодиться для управления разработкой. Подробности приведены в Get Viewer API (*Get Viewer*). Если параметр опущен, возвращается самая свежая версия файла.

Сначала создайте новый пустой документ html с элементами head и body. Добавьте следующие ссылки в элемент head, чтобы присоединить необходимые стили и JavaScript для viewer:

```
<link rel="stylesheet" href="https://developer.api.autodesk.com/viewingservice/v1/viewers/style.css" type="text/css">
<script src="https://developer.api.autodesk.com/viewingservice/v1/viewers/viewer3D.min.js"></script>
```

Теперь создайте элемент <script> и добавьте функцию инициализации вьюера initialize():

```
function initialize() {
  var options = {
    'document' : 'urn:dXJuOmFkc2sub2JqZWN0czpvcy5vYmplY3Q6bXlidWNrZXQvc2t5c2NwcjEuM2Rz',
    'env':'AutodeskProduction',
    'getAccessToken': getToken,
    'refreshToken': getToken,
  };
  var viewerElement = document.getElementById('viewer');
  var viewer = new Autodesk.Viewing.Viewer3D(viewerElement, {});
  Autodesk.Viewing.Initializer(options,function() {
    viewer.initialize();
    loadDocument(viewer, options.document);
  });
}
// This method returns a valid access token  For the Quick Start we are just returning the access token
// we obtained in step 2.  In the real world, you would never do this.
function getToken() {
  return "GX6OONHlQ9qoVaCSmBqJvqPFUT5i";
}
```

В этом примере параметр options.document – это URN в кодировке base64 для зарегистрированного объекта, который мы создали ранее. Для вашей модели этот параметр будет, разумеется, иным.

Функция getToken() в этом коротком примере просто возвращает токен доступа, представленный в ее коде в явном виде. В реальном приложении вы НИКОГДА не будете использовать такое решение как небезопасное. Функция, передаваемая в loadDocument() может иметь любую реализацию, если она предоставляет действительный токен аутентификации. Например, она может обратиться к вашему собственному веб-сервису, который в свою очередь вызывает Authentication API. Вам не следует напрямую вызывать Authentication API, т.к. это раскрывает для потенциального злоумышленника ваши Client ID и Client Secret.

Пример реализации простого сервера для получения токенов доступа «AuthTokenServer_Simple token server example» вы можете найти на GitHub: https://github.com/Developer-Autodesk/AuthTokenServer_Simple. Это может существенно упростить разработку клиентской части.

Вот как может выглядеть функция getToken(), если вы используете AuthTokenServer_Simple:

```
function getToken() {
    var theUrl = "http://localhost:5000/auth"; // change this when deploying
    var xmlHttp = null;
    xmlHttp =new XMLHttpRequest();
    xmlHttp.open("GET", theUrl, false);
    xmlHttp.send(null);
    var resp =  JSON.parse(xmlHttp.responseText);
    var token = resp["access_token"];
    return token;
}
```

Добавьте вторую функцию для управления загрузками и обработки сообщений об ошибках, когда документ загружен:

```
function loadDocument(viewer, documentId) {
  // Find the first 3d geometry and load that.
  Autodesk.Viewing.Document.load(documentId, function(doc) {// onLoadCallback
  var geometryItems = [];
  geometryItems = Autodesk.Viewing.Document.getSubItemsWithProperties(doc.getRootItem(), {
      'type' : 'geometry',
      'role' : '3d'
  }, true);
  if (geometryItems.length > 0) {
      viewer.load(doc.getViewablePath(geometryItems[0]));
  }
}, function(errorMsg) {// onErrorCallback
  alert("Load Error: " + errorMsg);
  });
}
```

Наконец, добавьте тело <body> как показано ниже:

```
<body onload="initialize()">
  <div id="viewer" style="position:absolute; width:90%; height:60%;"></div>
</body>
```

Откройте файл в поддерживаемом браузере, добавив ваш токен доступа к URN, как показано ниже (вы должны использовать локальный веб сервер):

```
http://localhost:8080/quickstart_3d.html?accessToken=GX6OONHlQ9qoVaCSmBqJvqPFUT5i
```

Теперь вы просматриваете ваш объект в Autodesk View and Data API Viewer!