Global Service Account (GSA) Authentication Documentation for Enterprise API Apps

v1.4 – Last Updated June 22nd, 2021

Contents

GSA Workflow for Enterprise API Applications

The NextGen API platform became capable of supporting the new **Global Service Account** (or "**GSA**") method of API authentication on November 11th, 2019.

All NextGen API Development Partners are required to use the GSA API authentication method (as opposed to any past API auth methods) when *any of the following* criteria apply:

- Integration of an API Application within the NextGen Enterprise environment in the NextGen API Test & Demo Suite (or "TDS") sandbox assigned to your organization *(The TDS environments are used only by API Partners who are 3rd Party Vendors).*
- Integration with *any* NextGen Client's NGE environment *(this criterion applies to all API Developers; NGE Client API partners who are self-developing apps, 3rd Party Vendor API Partners who are ready to integrate with a mutual client for API app beta testing, or otherwise).*

The benefits of implementing this new GSA API Authentication method include:

- Streamlining the mutual client onboarding experience (eliminating manual processes required of client admins & individual users);
- Improving security & scalability of the platform;
- Eliminating enforcement of end user credential strength policy as an API prerequisite.

The GSA method is still based on OAuth2, so your apps will still need to obtain a Bearer token that will be required to access Enterprise API routes beyond the initial OAuth route. Instead of using NextGen user credentials, you will use your **client_id** & **client_secret** *(assigned to your app in the Developer Portal)* together with a new client-side parameter called **site_id**.

The **site_id** value uniquely identifies a NextGen Enterprise (or "NGE") environment (*see Important Notes below*); your app will specify the **site_id** during GSA API authentication – the site_id value determines which unique NextGen installation (TDS sandbox, Beta Client TEST environment, or otherwise) your app will authenticate into and subsequently call for API data transactions.

- ***Vendor API Partners***: Instructions for locating the **site_id** of your assigned TDS environment are included in the TDS guidance document provided to the users in your organization who are authorized for TDS access.
- ***Important Notes Regarding Site_IDs:***
  - Each unique **site_id** corresponds to a unique NGE *environment*.
  - A NGE environment may contain one or more Enterprises (which in turn contain Practices).
  - A NGE Client will typically have at least two environments (e.g. TEST & PROD).

Important Notes Regarding Site_IDs, cont.

- If your API app authenticates via this new GSA method using the TEST site_id, the app's API calls will be made against the Enterprises/Practices/etc. in the TEST environment database.
- If your API app authenticates using the client's PROD site_id, then the app will be running against the client's live production NextGen system.
- **Therefore, it is <u>crucial</u> to use the correct site_id to avoid calling a Production environment during development, testing, or Beta phases.**

➢ NGE Clients will provide their TEST site_id to you when your app is approved for Beta by NextGen and that Client.

➢ The PROD site_id will be provided by Clients once your app is approved for Production release by NextGen and that Client.

Once a token is returned, your app is then required to:

1. Set the **Login Defaults** for the Global Service Account via the PUT /users/me/login-defaults route.
    a. The Login Defaults represent the following identifiers for your API App's initial session:
        i. Enterprise ID
        ii. Practice ID
    b. Setting the enterpriseId & practiceId via the PUT request will return a X-NG-SESSIONID representing the login defaults in all subsequent requests.
2. Using the token & login defaults (the latter represented in your X-NG-SESSIONID value), proceed to call data endpoints as per your app workflows.

Please review the **GSA Sequence Diagram** and detailed request examples on the following pages.

Note: A small minority of Enterprise API routes will require **Extended Login Defaults** – the vast majority of developers do not need to concern themselves with this process, but Extended Login Defaults are required for a handful of routes. If this is the case, routes presented with a X-NG-SESSIONID only representing enterpriseId & practiceId will respond with a message indicating that additional login defaults are required. If this occurs, you will need to create a new X-NG-SESSIONID to represent these additional login-defaults. Due to the additional steps of creating Extended Login Defaults and their general inapplicability to most Enterprise API routes.

Extended Login Defaults are also be represented by the X-NG-SESSIONID value. Creation of an X-NG-SESSIONID for Extended Login Defaults occurs via the same process as regular Login Defaults, but the following 5 IDs must be provided in the body of the PUT /users/me/login-defaults request:

- Enterprise ID
- Practice ID
- Provider ID
- Location ID
- Provider's Time Zone

See the GSA Authentication: Extended Login Defaults Sequence Diagram for a visual depiction of this process.

This document includes:

- The steps to configure and authenticate an Enterprise API App via /token using the GSA method
- How to configure a session via the **PUT** /login-defaults route to return a valid **X-NG-SESSIONID** header, which is returned upon successful initial and final setup of the login-defaults route.

Once the access token is used with the X-NG-SessionID header value, a valid response will be sent for all* subsequent requests. (*only routes that are approved for GSA will function. Reference the GSA tab of the Enterprise API Data Grid (available within www.nextgen.com/api) to review GSA availability status of any routes required by your app).

Preparing your API App for GSA in the Developer Portal

You must first add the **Enterprise API** to your app in the Developer Portal before GSA authentication will work correctly.

1. Your Org Admin will begin this task by logging into the Developer Portal.
2. Detailed instructions for adding the required API to your app are provided in the [Getting Started Wiki] > [Introduction to Developer Portal & NextGen APIs] page in the Wiki section of the Developer Portal.

How to Make Enterprise API Calls

Enterprise APIs allow you to GET, PUT, POST, PATCH and/or DELETE delete data from a practice's NextGen® Enterprise EHR (formerly known as NextGen® Ambulatory EHR) and NextGen® Enterprise PM (formerly known as NextGen® Practice Management) applications.

Note: Enterprise APIs are versioned to indicate the version of the EHR that the APIs work with. Making authentication calls as described in the steps below is not required to view the various API details and documentation in the API Explorer tool of the Developer Portal. **GSA Authentication is automated in the Enterprise API Postman Sandbox Collection** provided in the Getting Started Wiki's Authentication Details page.

To make Enterprise API calls, perform the following procedure:

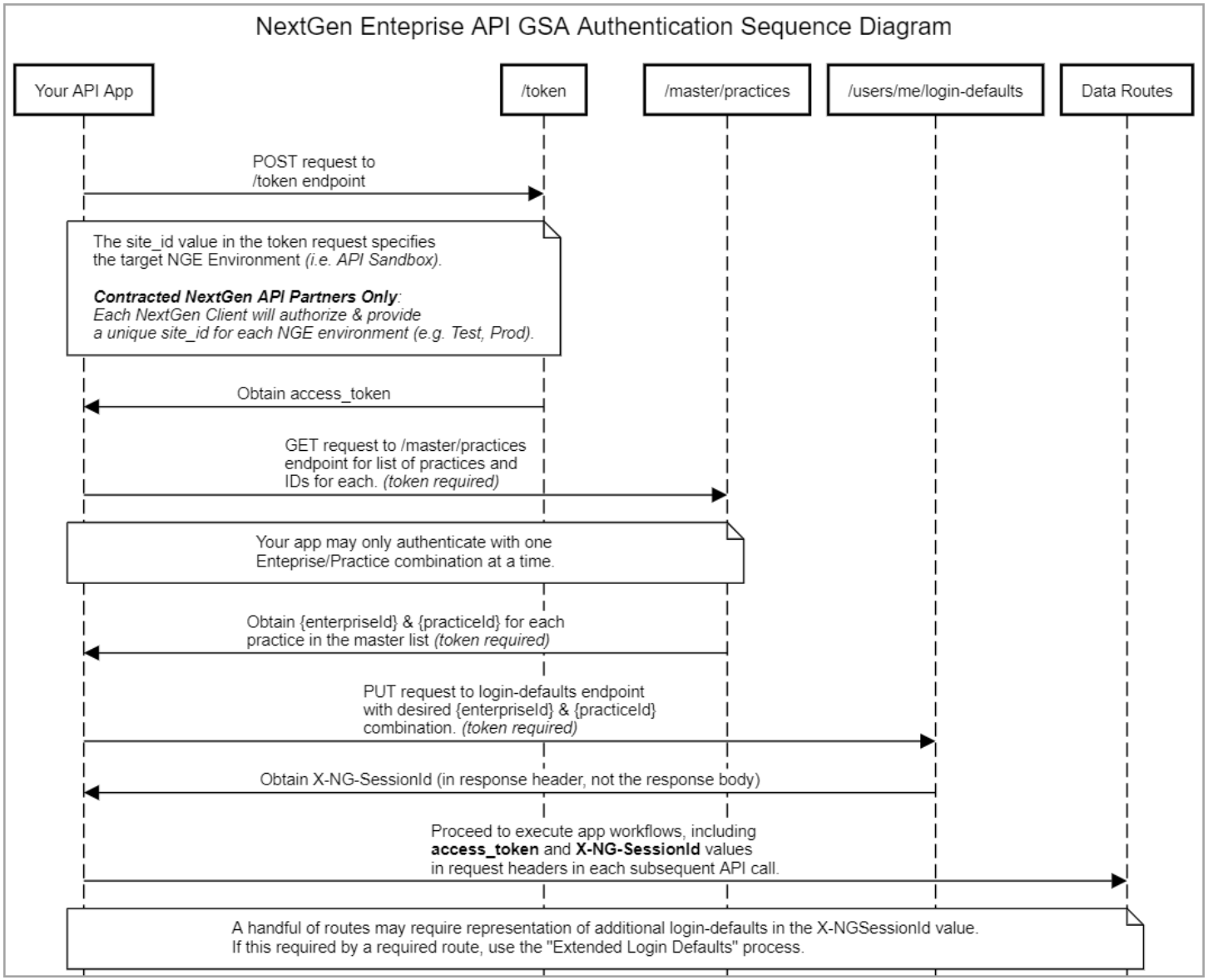
1. From your application, make an HTTPS POST request to the GSA Authentication /token endpoint detailed below using the Client Credentials grant_type.

Important: GSA Authentication requires your app to use *URLs that are different from the ones documented for the public API sandbox* in the Developer Portal. The GSA method requires you to use the following base URLs to construct endpoints when calling authentication & data routes, respectively:

- **API endpoint for GSA Authentication**:
  - GSA OAuth2 Base URL: `https://nativeapi.nextgen.com/nge/prod/nge-oauth`
  - GSA /token Endpoint: `https://nativeapi.nextgen.com/nge/prod/nge-oauth/token`
- **API endpoint for Data Route Calls under a GSA-Authenticated Session:**
  - Base URL: `https://nativeapi.nextgen.com/nge/prod/nge-api/api`
  - Example Route Path: `/users/me/login-defaults`
  - Endpoint: `https://nativeapi.nextgen.com/nge/prod/nge-api/api/users/me/login-defaults`

2. Obtain and use access_token in headers as Authorization: Bearer {access_token_value} in all subsequent routes. **Tokens expire in 1 hour.** ***Refresh tokens are not supported under GSA Authentication.***
3. Set login defaults and represent them as the value of a X-NG-SessionId header in all subsequent routes as per the sequence diagrams and documentation on the following pages.

GSA API Authentication Summary Diagram



OAuth2 Authentication

Enterprise API (Enterprise API Versions 5.9.0, 5.9.1, 5.9.2, 5.9.3, 5.9.4, & 6.0.0):

- Method: **POST**
- Token URL: https://nativeapi.nextgen.com/nge/prod/nge-oauth/token
    - ***Note***: Only the /token endpoint is used to authenticate; the OAuth2 /authorize endpoint is not used with GSA.

Sample Authorization Call (curl format):

```
curl -X POST \
  'https://nativeapi.nextgen.com/nge/prod/nge-oauth/token?grant_type=client_credentials&client_id=YOUR_CLIENT_ID&client_secret=YOUR_CLIENT_SECRET&site_id=YOUR_NEXTGEN_CUSTOMER_API_SITE_ID' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
```

POST https://nativeapi.nextgen.com/nge/prod/nge-oauth/token

Token Request – Required Query Parameters:

grant_type	client_credentials
client_id	Issued upon NG Admin approval of App in NextGen Developer Portal
client_secret	Issued upon NG Admin approval of App in NextGen Developer Portal
site_id	NextGen Global Service Account site_id

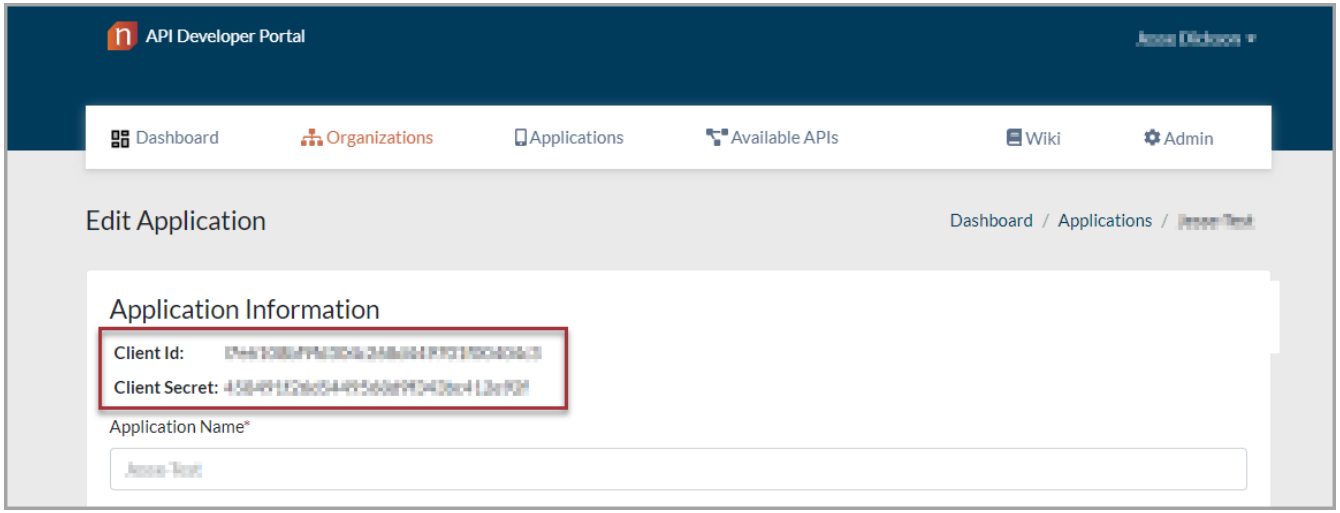
Enterprise API Global Service Authorization Token Request

Header	**Value**
Content-Type	application/x-www-form-urlencoded
Response	*Value*

Example Response:	{
	"access_token": "c1c44a7e-581a-4cc1-8a98-bad22f3b0d59",
	"token_type": "Bearer",
	"expires_in": 3600,
	"scope": "oob"
	}

Access_token	{{access_token}}
Token_type	Bearer
Expires_in	3600
Scope	oob

Application **client_id** and **client_secret** are both displayed in the [Applications] > [**Edit Application**] screen within the Developer Portal as shown here:



API Call following /token to PUT /login-defaults for the Global Service Account:

The primary purpose of login defaults is to allow you to set the values once per session rather than send these values on every single call. Most routes at minimum require the enterprise & practice IDs to be supplied. Others will in addition require the timeZone, locationId and/or providerId to be set as well. A call to set the login defaults will return a header value named **X-NG-SessionId**

The X-NG-SessionId MUST be included within all subsequent API requests using. If utilizing routes such as POST /encounter, when the rendering provider or location Id is not defined in the request body, then the login-defaults provider id and location id would be used. If the location id and provider Id were not set in the login defaults some routes will not work.

If you make any further API calls and pass this value X-NG-SessionId in the request headers, the login-defaults values that were passed will be used for that request regardless of whether or not any further calls were made to set the login-defaults. This allows multiple sessions to use the same credentials and different login-defaults.

An application will at minimum establish a X-NG-SessionId representing an enterprise and practice ID combination. A X-NG-SessionId representing enterprise & practice IDs is also necessary to obtain the subsequent provider, location and time zone name to establish Extended Login Defaults.

To set the login defaults for the currently logged in user, you would call the route /users/me/login-defaults

PUT https://nativeapi.nextgen.com/nge/prod/nge-api/api/users/me/login-defaults	
Authorization Header	**Value**
Bearer_token	{{access_token}}
Header	**Value**
Content-Type	application/json
Request Body	**Value**
enterpriseId	String
practiceId	String

JSON Request body for PUT login-defaults example **with practiceId & enterpriseId:**

```
{
“enterpriseId”: “00001”,
“practiceId”: “0001”
}
```

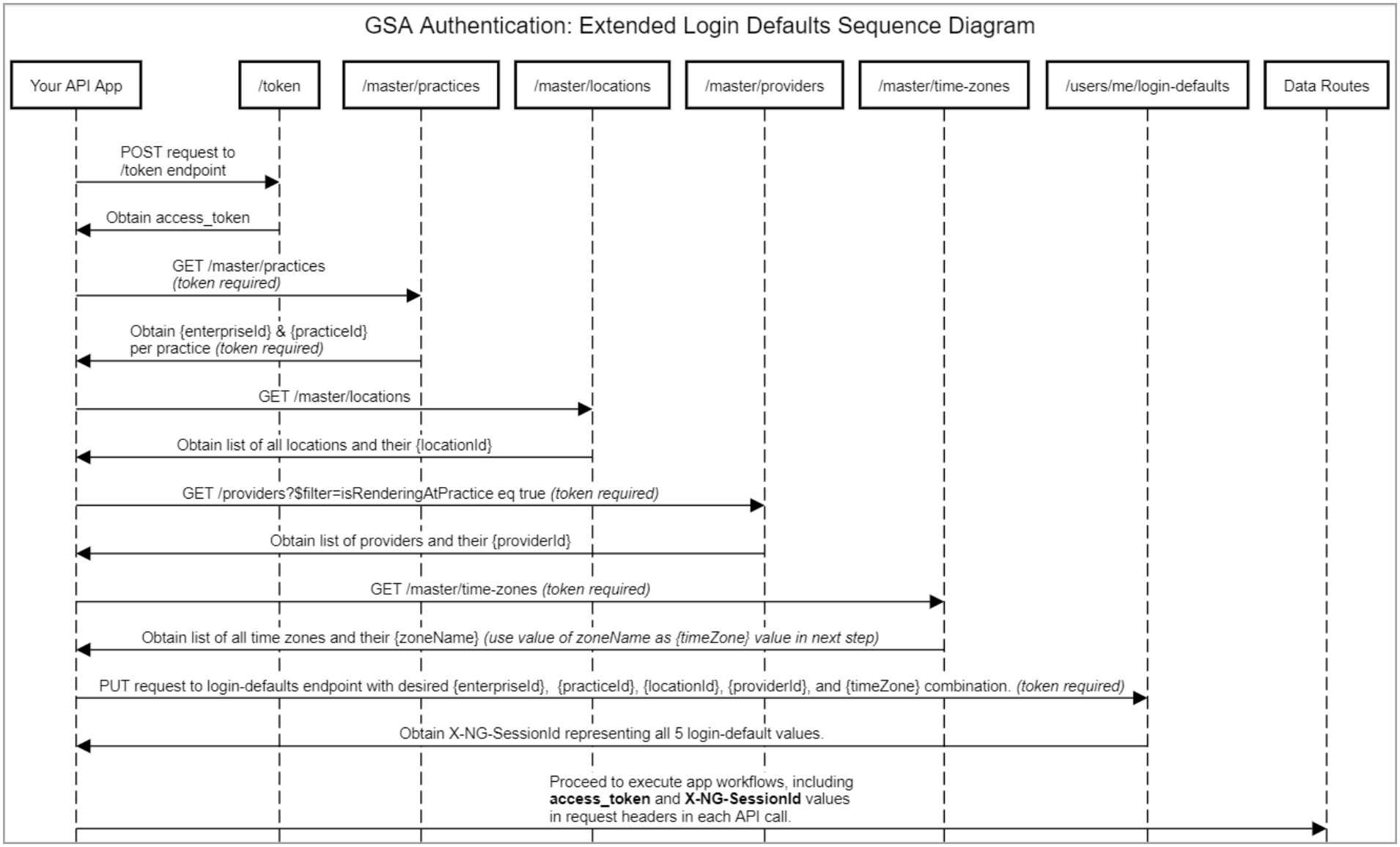
(CURL example): Login-Default with only Enterprise and Practice ids to return X-NG-SessionId as a response header

```
curl -X PUT \
        https://nativeapi.nextgen.com/nge/prod/nge-api/api/users/me/login-defaults \
        -H 'Authorization: bearer 4cf01b38-d9e6-4402-8b7a-b4628ee7ac07' \
        -H 'Content-Type: application/json' \
        -H 'cache-control: no-cache' \
       {
      “enterpriseId”: “00001”,
      “practiceId”: “0001”
       }'
```

Response *header* from PUT login-defaults returns X-NG-SessionId

Response header	Value
X-NG-SessionId	{{sessionId}}

Sequence Diagram & Examples for Setting Extended Login Defaults



Lookup Provider ID for the logged in practice

To set the login defaults with a valid provider id for the currently logged in user, the request to return a list of the available provider id's would be called with the X-NG-SessionId included in the request. An application would want to set a login-default with a valid provider for the NextGen practice they are implementing their application with. This would be verified with the on-boarding NextGen client which provider id can be setup for the Global Service Account user.

GET /master/providers can be filtered to return a list of the valid rendering providers for the current practice setup within the login defaults initial request.

GET https://nativeapi.nextgen.com/nge/prod/nge-api/api/providers?$filter=isRenderingAtPractice eq true

Authorization	**Value**
Bearer_token	{{access_token}}

Header	**Value**
X-NG-SessionID	{{session_Id}}

(curl example) Master Providers route with X-NG-SessionID

In the example below, the route is setup to filter out non-rendering providers from the response. This is accomplished with the filter setup to excluded non-rendering providers by filtering using the $filter=isRenderingInPractice eq true:

```
curl -X GET \
  'https://nativeapi.nextgen.com/nge/prod/nge-api/api/providers?$filter=isRenderingAtPractice
eq true '\
  -H 'Accept: */*' \
  -H 'Authorization: Bearer 7a760943-abaa-4d1f-a354-e0e22df79ffa' \
  -H 'Content-Type: application/json' \
  -H 'X-NG-SessionID:
MDAwMDF8MDAwMXwwMDAwMDAwMDAwMDAwMDAwMDAwM
DAwMDAwMDAwMHwwMDAwMDAwMDAwMDAwMDAwMDAw
MDAwMDAwMDAwMDAwMHwzMzczMw==' \
```

Response example: Providers for the Enterprise and Practice are returned. The Provider ID can be added to a new PUT login-defaults request in a subsequent step.

Important Note: The **id** in "id": "*[value]*" is the **providerId** in the response example from the /providers route below – please ensure your app appends the proper prefix (in this case, 'provider'). The Id of items returned in the list response from GET /providers is returned simply as "**id**", but Enterprise API will expect you to use **providerId** when using these Ids in other routes.


```
{
  "items": [
    {
      "id": "f725ac67-d666-4b35-8bd3-0648643a560a",
      "lastName": "Jones",
      "firstName": "Brian",
      "middleName": "",
      "description": "Jones, MD Brian",
      "providerAddressLine1": "14 Main Street",
      "providerAddressLine2": "",
      "providerCity": "Horsham",
      "providerState": "PA",
      "providerZip": "19044",
      "providerPhone": "2154135135",
      "providerPhoneExtension": "",
      "isDeleted": false,
      "degree": "",
      "enterpriseId": "00001",
      "practiceId": "0001",
      "isReferringProvider": true,
      "isRenderingAtPractice": true,
      "isSupervisingProvider": false,
      "language1Id": null,
      "specialtyCode1": "",
      "providerSubgrouping1Id": null
}
```

Lookup Location ID for the logged in practice

To set the login defaults with a valid location id for the currently logged in user, you would call the route for master/locations with $filter=isSchedulable is true and isDeleted is false.

```
GET https://nativeapi.nextgen.com/nge/prod/nge-api/api/master/locations?$filter=isDeleted eq false and isSchedulable eq true
```

Authorization	Value
Bearer_token	{{access_token}}

Header	Value
X-NG-SessionID	{{session_Id}}

(curl example) GET /master/locations route with X-NG-SessionID

```
curl -X GET \
  'https://nativeapi.nextgen.com/nge/prod/nge-api/api/master/locations?$filter=isDeleted eq
false and isSchedulable eq true' \
  -H 'Content-Type: application/json' \
  -H 'X-NG-SessionId:
MDAwMDF8MDAwMXw5ZDk3MWU2MTJiNWE0NTA0OTAxNjdmZDg2Mzc5MGVlMnxjNzA2YWM
5YjA3YTU0NDM2OWJkMDk5YWMwNGFmNWNlMnwzMzc5NA==' \
  -H 'Authorization: Bearer 7a760943-abaa-4d1f-a354-e0e22df79ffa' \
```

Response example for GET /master/locations: Locations for the Enterprise and Practice are returned. The location id can be added to a new PUT login-defaults request in a subsequent step.

Important Note: The **id** in "id": "*[value]*" is the **locationId** in the response example from the /master/locations route below – please ensure your app appends the proper prefix (in this case, 'location'). The items returned in the list response from GET /master/locations are uniquely identified via the "**id**" value, but Enterprise API will expect you to pass these values using **locationId** when designating locations in other routes.


```
"items": [
 {
  "id": "a92974dd-c694-46ea-b8ad-05888f7b5262",
  "name": "Intake - Men",
  "isDeleted": false,
  "addressLine1": "398 Atlanta Rd",
  "addressLine2": "",
  "city": "Atlanta",
  "state": "GA",
  "zip": "30033",
  "defaultLabId": null,
  "defaultRadiologyId": null,
  "defaultRegistryId": null,
  "isSchedulable": true,
  "_links": [
```

Lookup Time Zone Name for the logged in practice

To set the login defaults with a valid UTC timezone for the currently logged in user, you would call the route for master/time-zones

```
GET https://nativeapi.nextgen.com/nge/prod/nge-api/api/master/time-zones
```

To filter GET time-zone by zoneName, use the $filter expression the OData filter `startswith` and `top` functions also work well here – i.e. $filter=startswith(zoneName, 'America')&top=100

```
https://nativeapi.nextgen.com/nge/prod/nge-api/api/master/time-zones?$filter=startswith(zoneName, 'America')&$top=100
```

Authorization	Value
Bearer_token	{{access_token}}

Header	Value
Content-Type	application/json
X-NG-SessionID	{{session_Id}}

(curl example) master/time-zones route with X-NG-SessionID

```
curl -X GET \
  'https://nativeapi.nextgen.com/nge/prod/nge-api/api/master/time-zones?$filter=startswith%28zoneName,%20%27America%27%29&$top=100' \
  -H 'Authorization: Bearer 374e3862-4fg7-4df1-a0ad-6f51a3d31dc9' \
  -H 'X-NG-SESSIONID:
MDMwMDF8MAZwMXw5ZDk3MWU2MTJiNWE0NTA0OTAxNjdmZDg2Nzc5MGVlMnwyM2U5MTdmZWRmMGM0ZDc1ODE3NzBiNzcxMzBjNDJiY3xBbWVya
VNhL05ld19Zb3JrfDEyNjAq' \
```

Response example for master/time-zones:


```
" {
"items": [
 {
  "zoneName": "America/Los_Angeles",
  "utcOffset": -28800,
  "utcOffsetDisplay": "UTC-8:00",
  "_links": [
   {
    "href": "/master/time-zones?$filter=ZoneName eq 'America/Los_Angeles'",
    "method": "GET",
    "rel": "self",
    "description": "Gets time zones",
    "vendorExtensions": {
     "baseHref": "/master/time-zones?$filter=ZoneName eq '{0}'"
    }
   }
```

Set Extended Login Defaults for Global Service with Enterprise Id and Practice Id, Provider Id, Location Id and Time-zone

To set “extened” login defaults using GSA, call the PUT /login-defaults route:

PUT https://nativeapi.nextgen.com/nge/prod/nge-api/api/users/me/login-defaults

Authorization	Value
Bearer_token	{{access_token}}

Header	Value
Content-Type	application/json

Request Body	Value
EnterpriseId	String,
PracticeID	String,
locationId	guid,
ProviderID	guid,
timeZone	String (Use desired zoneName value from a /master/time-zones response)

Request body example

```
{
        "enterpriseId": "00001",
        "practiceId": "0001",
        "locationId": "9e8eb554-e636-4cd3-b68f-86d21434cb72",
        "providerId": "46c7a9ea-0b7a-483a-9955-0f5cf66e3b7b",
        "timeZone": "America/New_York"
}
```

(CURL example): Setting Extended Login Defaults to return X-NG-SessionId as a response header:

```
curl -X PUT \
        https://nativeapi.nextgen.com/nge/prod/nge-api/api/users/me/login-defaults \
        -H 'Authorization: bearer 4cf01b38-d9e6-4402-8b7a-b4628ee7ac07' \
        -H 'Content-Type: application/json' \
{
       "enterpriseId": "00001",
       "practiceId": "0001",
       "locationId": "9e8eb554-e636-4cd3-b68f-86d21434cb72",
       "providerId": "46c7a9ea-0b7a-483a-9955-0f5cf66e3b7b",
       "timeZone": "America/Los_Angeles"
        }
```

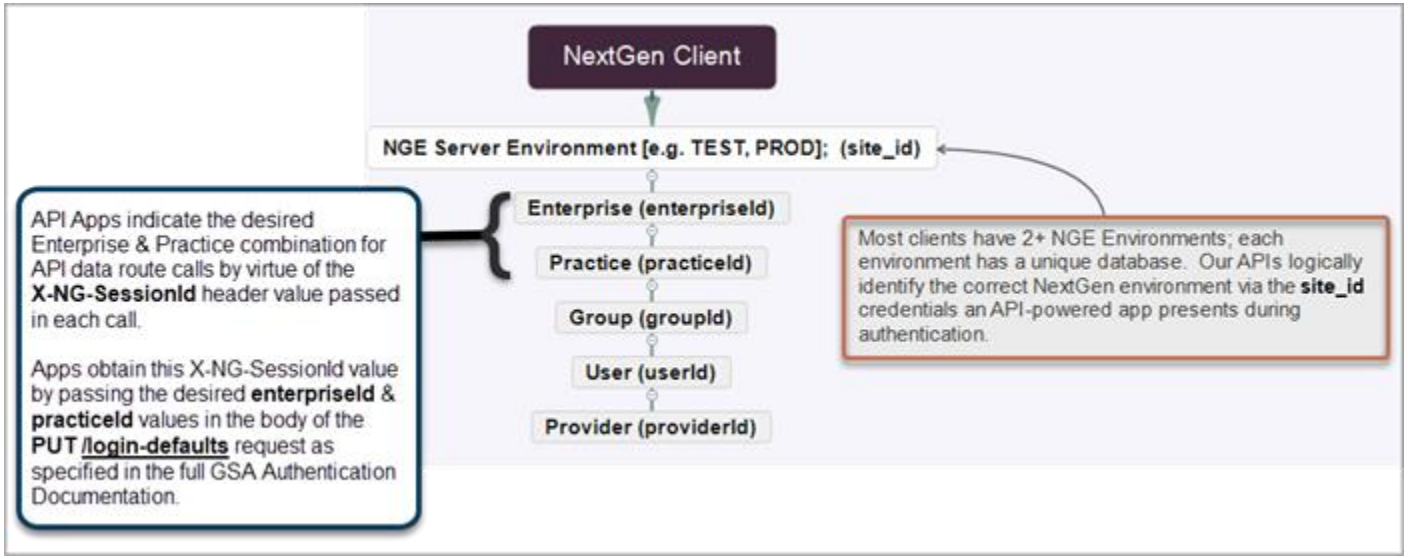
Response header	value
X-NG-SessionID	{{session_id}}

- The Location ID and Provider ID are unique per NextGen Database (Test, Prod) and Practice, so when authenticating in a multiple enterprise or multiple practice implementation, **the site_id will be different for Test and Production**.
- Each database and practice will have a unique X-NG-SessionId. The login defaults are required to be configured for each of the NextGen customers and the X-NG-SessionId will be unique per Enterprise and Practice, but the site_id will only be unique per database.

- Enterprise API access_tokens are issued via the GSA OAuth2 client_credentials grant and expire after 1 hour.
    - The refresh token flow is not supported – thus we recommend you execute a timed update script or similar approach when expiry approaches (e.g. ~55m) to ensure your app is always using a fresh token.
    - Tokens are granted at the NextGen Enterprise Server Environment level (see Figure 1 on next page), meaning a token issued using the client's API site_id credential for their Test environment will force your app to communicate with that Test environment. The same is true for the Production site_id/env.
    - If your app is meant to integrate with more than once NextGen Enterprise® EHR Client, you will need to implement a method enabling your app to obtain, update, and use an access_token that corresponds with each Client's site_id whenever your app is communicating with that Client's NGE system.
    - The token is valid for use to integrate with any of the Enterprise(s) and Practice(s) within a given client NGE system, however do not assume each NextGen client desires integration with any & all available Enterprise(s) & Practice(s) in their system – it is important you discuss the scope of integration with each client in terms of their desired Enterprise(s) & Practice(s) to ensure your app only communicates with those that are approved by each client.
- With a valid token, you can then call the GET /master/practices route to obtain the names of the practices within the NGE environment and the corresponding {enterpriseId} and {practiceId} for each (there's no need to first call GET /master/enterprises since the {enterpriseId} is included in each result from GET /master/practices).
- With knowledge of an enterpriseId & practiceId combination, you call PUT /login-defaults.
    - The PUT /login-defaults response "body" only contains a status code, whereas the crucial X-NG-SessionId value will be returned within the response header.
    - The X-NG-SessionId is not a true jsessionid, but rather a 2nd "key" (the 1st being the token) needed to call data from a given Enterprise+Practice combination – X-NG-SessionId espouses the data you submit via the json body in your call to PUT /login-defaults.
    - X-NG-SessionId values are not dynamic, meaning they will remain constant given the same input values; they also do not expire – therefore you can store each client's set of one or more X-NG-SessionId values in a config file, etc. There is no need to update the X-NG-SessionId once you've established that value for a given combination of login-defaults (Exception: Certain personalization routes require additional elements in the PUT /login-defaults body; however the only values you'll typically need to PUT to /login-defaults are enterpriseId & practiceId).
    - These attributes of X-NG-SessionId have significant implications:
        - When using a valid token and a given X-NG-SessionId (each as header values) to subsequently call "data routes", the responses from those data routes will be limited to data from the specified enterpriseId & practiceId combination represented by the X-NG-SessionId value.

- If your app needs to obtain data from multiple practices, you must request that data one practice at a time, changing the X-NG-SessionId per request to target the {enterpriseId}+{practiceId} combination espoused by their corresponding X-NG-SessionId.
- It is likely far more efficient to execute a multi-route workflow under one X-NG-SessionId then re-execute the workflow for another practice by changing X-NG-SessionId (as opposed to calling the same route for each practice, varying the X-NG-SessionId each time – then proceeding to do the same for each subsequent route in a workflow).

Figure 1: The hierarchical structure of a NextGen Enterprise system relevant to GSA Authentication is shown below:



- Client NGE Installations can contain one or more NGE Server Environments (minimally Test & Prod).
  - Environments (e.g. “NGTest”, “NGProd”) have a truly unique API site_id.
  - Each environment can contain one or more Enterprises;
- Enterprises have a non-unique {enteprieseId} - non-unique meaning it is probable that most clients will have the same {enterpriseId} value(s) as every other client – the site_id is the one value that will remain truly unique at all times in a multi-client scenario; keep this in mind when organizing your approach.
  - Enterprises contain Practices. Some NGE environments have more than one Enterprise (so plan accordingly if this is a scenario you might encounter), but most environments will have but one Enterprise and in those cases all Practices will share the same enterpriseId value of 0001.

- Practices have a similarly non-unique {practiceId} (although multi-Practice Enterprises are far more common in the NextGen client base than multi-Enterprise Environments).
    - Practices contain Groups. Groups are important for Client-side NGE operations but are typically irrelevant in API development unless important for Tasking workflows.
- Groups contain Users. User Permissions are defined per Group by each Client via their NGE System Administrator application.
- Users inherit Permissions based on their membership within one or more Group.
- Providers are a subset of Users, and while all Providers are Users, not all Users are Providers.
    - Routes always requiring a {providerId}, {locationId}, and {timeZone} to be set via login defaults will indicate this requirement in an error response if you attempt to call them using an X-NG-SessionId value that only represents {enterpriseId} & {practiceId} values.

Questions about GSA Authentication

If you are uncertain of the access account method your app should use in Production, your App's route list and route GSA status, please reach out to NextGen via APIpartners@nextgen.com to discuss your options and determine the right approach for your app.