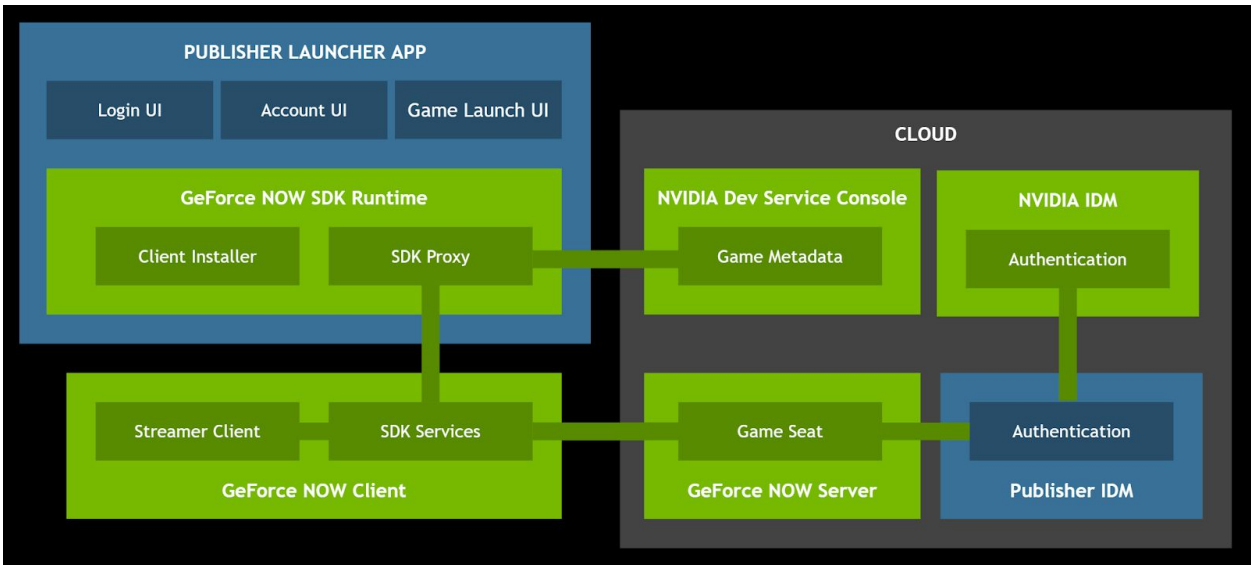
GFN Runtime Software Development Kit

Table of Contents

1. Introduction

The NVIDIA GFN Runtime SDK provides a set of interfaces to allow game developers and game publishers to interact with parts of the NVIDIA GeForce NOW ecosystem. Integration is provided by

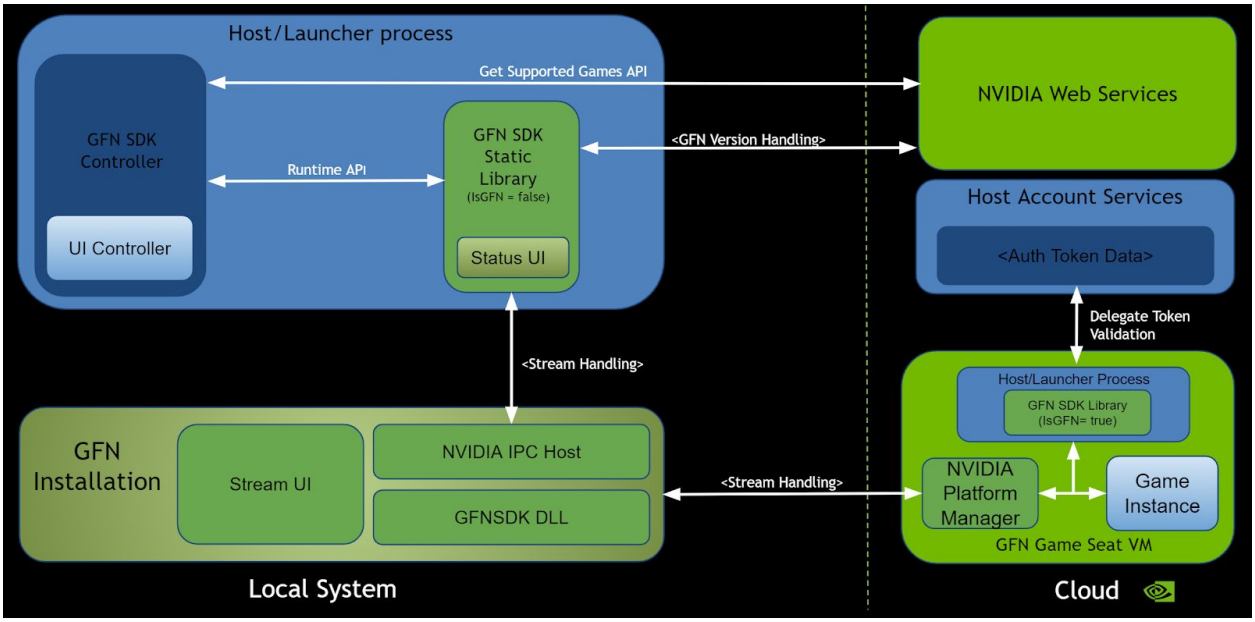
various means, from native C interfaces to RESTful Web API calls, depending on the feature of the SDK.



This document provides details of how to integrate the GFN Runtime SDK features into your application and its developer and deployment processes. For more detailed technical information about the architecture, refer to the architecture section later in this document.

2. C API for Launcher or Game Integration and Single Sign-On (SSO)

2.1. Architecture



The GFN Runtime SDK provides a static C library that is linked to the game/application and exposes the various APIs defined in this document. The behavior of those APIs depends on the environment

the application is running in; either on a client/user system or in the GeForce NOW (GFN) cloud environment.

On client systems, this library checks for the presence of the GeForce NOW (GFN) client installation at initialization time. If the client is not present, or the client is out of data, then the library will initiate a download and installation of the latest client on the first API call that requires the client to be present, presenting UI to the user for the installation process.

If the GFN client is installed on the system, the static library will defer API actions to components in the installation.

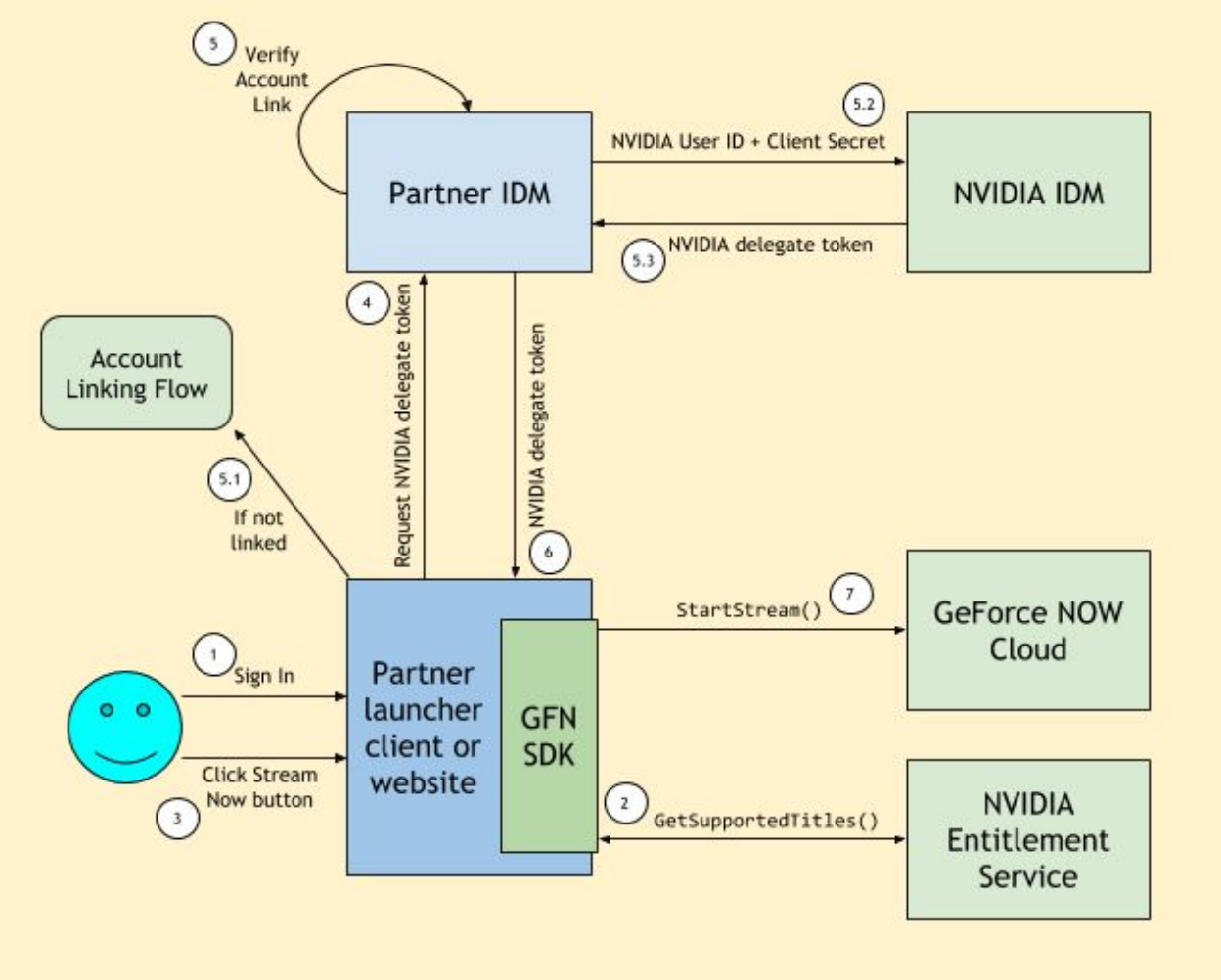
This design allows the static library to stay as thin as possible in the application, and provides backward and forward compatibility to new GFN client packages.

On GFN game systems, many of the APIs are no-ops as they apply only to client/end-user systems. In those cases, API calls will return a well-defined error code to denote the call was not applicable to the environment.

2.2 Authentication and Federation

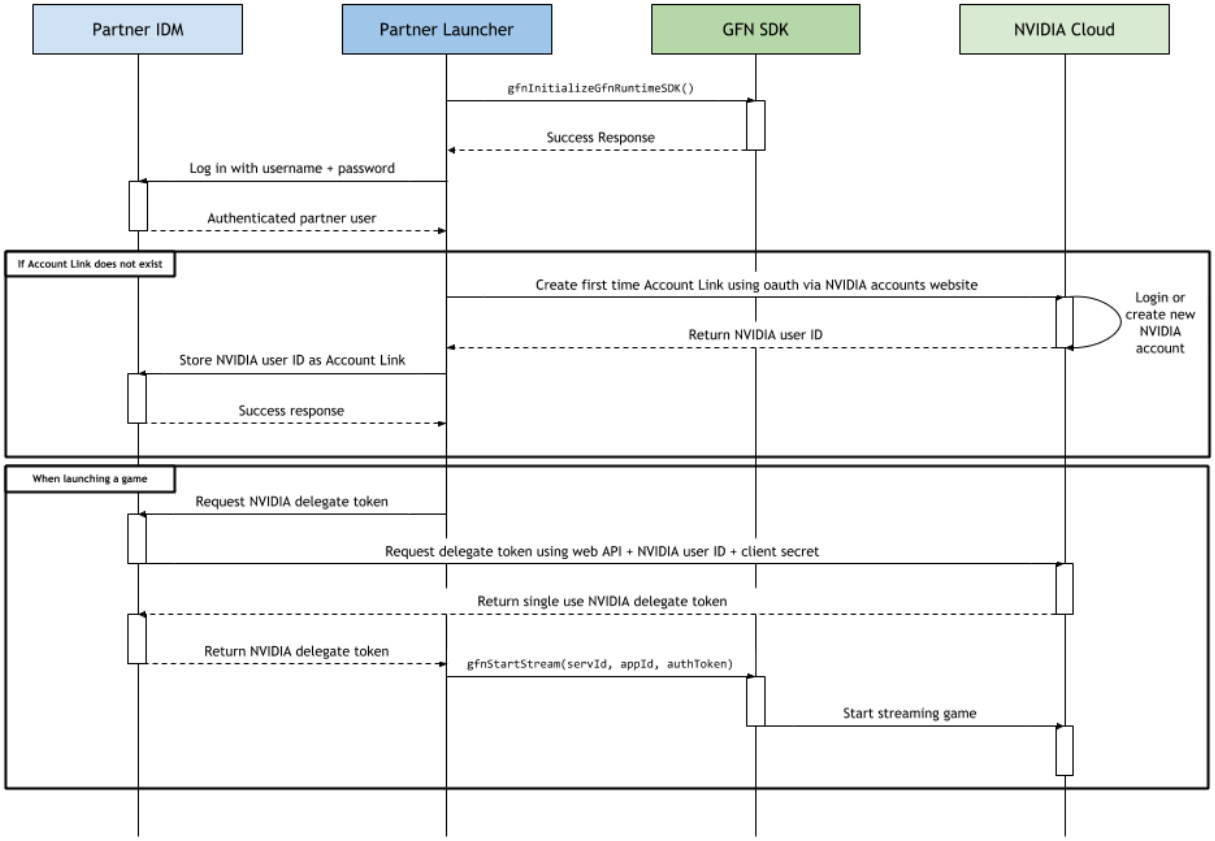
NVIDIA provides a backend IDM Service that application developers can use to validate users and obtain user information from. This provides a seamless flow for users to bypass multiple login steps in streaming a game through GFN.

A simplified flow diagram of how this functions is shown below:

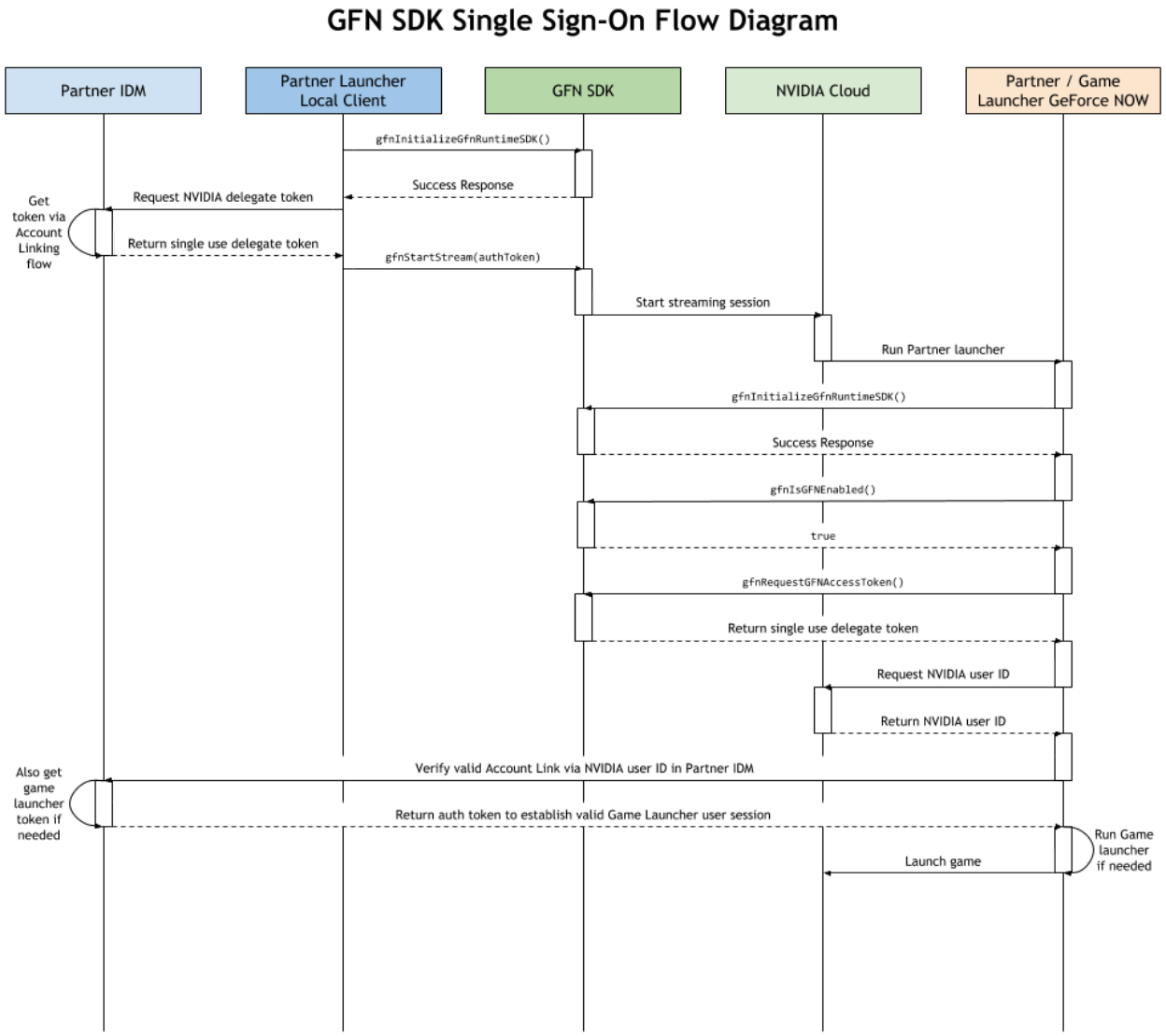


Authentication between the Partner IDM and NVIDIA IDM happens using secure HTTPS web API calls, and the account linking flow utilizes standard oauth2 protocol. Once the account link is established the authentication process between Partner and NVIDIA becomes transparent to the user, and gaming streaming can be initiated without requiring any further authentication or manual login.

GFN SDK Account Linking Flow Diagram



After the account link between Partner and NVIDIA has been established, that link can be utilized on the GFN server to facilitate Single Sign-On (SSO) so that the user does not have to manually login again, but all authentication happens transparently and the game launches immediately.



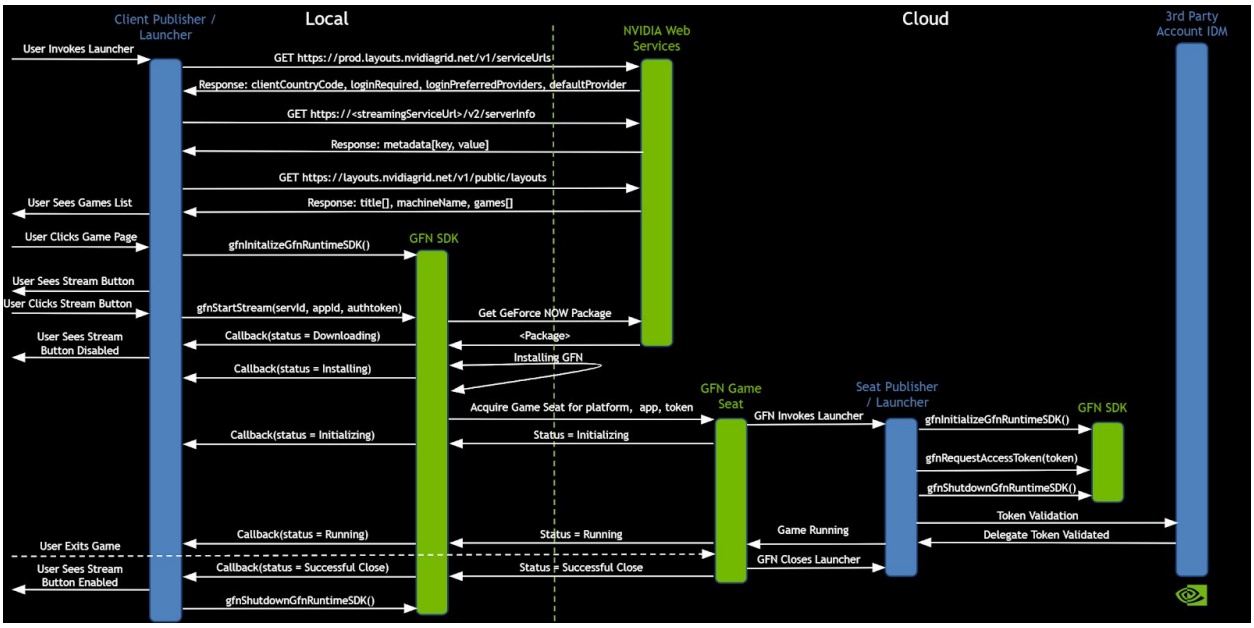
For additional high-level overview, please refer to the SDK primer available as part of the documentation section of the SDK's repository.

2.3. GFN Runtime API Overview

GFN Runtime API methods are used to make requests from or to notify the GFN backend.

When your application is operating outside of the GFN environment, these methods are simple stubs that incur almost no cost, so it's safe to add these to your main build.

The calling convention differs by which API you've selected to use. In most cases, the methods return a GfnRuntimeError result, which can be used by the application to check for errors. In addition, some methods are asynchronous by nature, but provide synchronous variants when possible.



2.4. General/Common Methods

InitializeGfnRuntimeSDK

```
C    GfnRuntimeError gfnInitializeGfnRuntimeSDK()
```

Description

Should be called at application startup and prior to any GFN Runtime API methods.

When running outside of a GFN environment (a game seat virtual machine or development environment) it is expected for this method to return a result other than success. In this case all GFN methods become no-ops and have no performance impact on your application.

Usage

Call as soon as possible during application startup.

Return values

GfnRuntimeError::gfnSuccess on success when running in a GFN environment.

GfnRuntimeError::gfnGfnDLLNotPresent if running outside a GFN environment (no GFN.dll present)

GfnRuntimeError::gfnGfnComNotEstablished if running outside a GFN environment (no GFN host or test application running)

GfnRuntimeError::gfnIncompatibleVersion Linked GFN Runtime SDK library is not compatible with the existing GFN.dll)

ShutdownGFNRuntimeSDK

```
C    void gfnShutdownGfnRuntimeSDK()
```

Description

Releases the SDK, and frees up memory allocated by GFN and disconnects from GFN backend.

Usage

Call during application shutdown or when GFN Runtime API methods are no longer needed.

2.5. Launcher Application-specific Methods

GetClientIp

C	GfnRuntimeError **gfnGetClientIp**(const char** ppchClientIp)

Description

Gets user's client IPv4 address

Since Applications running under GFN run in the GFN data centers, any IP queries made by the Application will return IPs associated with the data center, not the user's client IP. This allows the Application obtain the client IP in v4 format so that developers can make regional business decisions.

Usage

Call this during application start or from the platform client in order to get the user's actual client IP address.

Parameters

ppchClientIp: Output IPv4 in string format. Example: "192.168.0.1"

Return value

gfnSuccess: On success

Otherwise, appropriate error code

GetClientLanguageCode

C	GfnRuntimeError **gfnGetClientLanguageCode**(const char** ppchLanguageCode)

Description

Gets user's client language code in the form <lang>-<country> using the standard ISO 639 language codes and ISO 3166 country codes.

Usage

Call this during application start or from the platform client in order to get the user's language and country settings

Parameters

ppchLanguageCode: Language code as a string. Example: "en-us"

Return value

gfnSuccess: On success

Otherwise, appropriate error code

IsRunningInCloud

C	`bool gfnIsRunningInCloud()`

Description

Determines if calling application is running in GFN environment or not.

Usage

Use to enable any GFN environment specific application logic, for example, to block any calls to gfnStartStream(), or to call gfnNotifyStorageChange() if the function returns true.

Return value

true: Application is running on a game seat virtual machine or GFN test environment

false: Application is not running in a GFN Environment

RequestGfnAccessToken

C	`GfnRuntimeError gfnRequestGfnAccessToken(const char** ppchToken)`

Description

Request to obtain a user specific access token to allow access to the GFN backend service (IDM endpoint).

Usage

The access token provided can be used by the application's backend servers to validate the user and obtain user data from the GFN backend service. The GFN backend service provides an OAuth2 interface for validating users and retrieving data. See Account Federation section for more information.

Parameters

ppchToken: Populated with a user specific GFN access token.

Return value

gfnSuccess: On success

Otherwise, appropriate error code

StartStream

```
C    GfnRuntimeError gfnStartStream(char* pchPlatformId, char* pchPlatformAppId, char*
     pchAuthToken);
```

Description

Requests GFN client to start a streamed session of an application

Usage

Use to start a streaming session.

Parameters

pchPlatformId: Identifier of the launcher service, e.g. “Steam”

pchPlatformAppId: Identifier of the requested application to start streaming

pchAuthToken: Authentication token from NVIDIA IDM

Return value

gfnSuccess: On success

Otherwise, appropriate error code

2.6. Game/Cloud Application-specific Methods

RegisterExitCallback

```
C    GfnRuntimeError gfnRegisterExitCallback(void *pCbPointer(void*), void *pUserContext);
```

Description

Register an application callback with GFN to be called when GFN needs to exit the game.

Callback function signature is:

```
void func(void* pUserContext)
```

Where the pUserContext pointer points to the user context pointer provided by the user as an argument to the registration method (can be NULL).

Usage

Call to query register callbacks with GFN

Parameters

pCbPointer: Pointer to callback function

pUserContext: Pointer to user defined context object, which will be returned as a parameter to the callback (can be NULL).

Return value

gfnSuccess: On success

Otherwise, appropriate error code

RegisterPauseCallback

C	`GfnRuntimeError gfnRegisterPauseCallback(void *pCbPointer(void*), void *pUserContext);`

Description

Register an application callback with GFN to be called when GFN needs to pause the game on the user's behalf.

For Multiplayer games, it is recommended that this is implemented similar to a client disconnect.

Callback function signature is:

```
void func(void* pUserContext)
```

Where the pUserContext pointer points to the user context pointer provided by the user as an argument to the registration method (can be NULL).

Usage

Call to query register callbacks with GFN

Parameters

pCbPointer: Pointer to callback function

pUserContext: Pointer to user defined context object, which will be returned as a parameter to the callback (can be NULL).

Return value

gfnSuccess: On success

Otherwise, appropriate error code

RegisterSaveCallback

C	`GfnRuntimeError gfnRegisterSaveCallback(void *pCbPointer(void*), void *pUserContext);`

Description

Register an application callback with GFN to be called when GFN needs the application to save user progress.

It is recommended that this be implemented as an autosave if such a feature is supported by your application.

Callback function signature is:

```
void func(void* pUserContext)
```

Where the pUserContext pointer points to the user context pointer provided by the user as an argument to the registration method (can be NULL).

Usage

Call to query register callbacks with GFN

Parameters

pCbPointer: Pointer to callback function

pUserContext: Pointer to user defined context object, which will be returned as a parameter to the callback (can be NULL).

Return value gfnSuccess: On success

Otherwise, appropriate error code

RequestKeyboardOverlayClose

C	`GfnRuntimeError gfnRequestKeyboardOverlayClose()`

Description

Called from application when necessary text input has been processed and user can continue. This would cause a previously requested keyboard overlay on the GFN user's client display to be dismissed

Usage

RequestKeyboardOverlayOpen should be called before this method. If not, RequestKeyboardOverlayClose will have no effect.

Return value

gfnSuccess: On success

Otherwise, appropriate error code

RequestKeyboardOverlayOpen

C	`GfnRuntimeError gfnRequestKeyboardOverlayOpen(GfnScreenPosition gspPosition)`

Description

Called from application when it is expecting text input from user. Calling this API would trigger a native keyboard overlay to be shown to the GFN user such that he/she can most easily enter text, based on the particular GFN client platform being used.

There's no special input handling needed from application; input will be injected into application by GFN (as is done in all other times running in GFN). Note GFN is not displaying any text input box or prompt to user, only a keyboard overlay.

Usage

This API should be called as a pair with `RequestKeyboardOverlayClose`. Multiple calls to `RequestKeyboardOverlayOpen` will have no effect after the first call.

Parameters

gspPosition: the desired screen positioning of text input element (i.e. Android keyboard). Should be one of the following values:

gspBottom

gspTop

gspLeft

gspRight

gspCenter

gspTopLeft

gspTopRight

gspBottomLeft

gspBottomRight

Return value

gfnSuccess: On success

Otherwise, appropriate error code

SetupTitle

C	`GfnRuntimeError gfnSetupTitle(char* pchPlatformId, const char* pchPlatformAppId)`

By default, should be async

Description

Notifies GFN that an application should be readied for launch. GFN will install the application by attaching its virtual drive, set all necessary settings to optimize for GFN and download user save data.

Will block until application setup is complete and the application is ready for launch.

Usage

Use prior to launching an application.

Parameters

pchPlatformId: Identifier of the launcher service, e.g. “Steam”

pchPlatformAppId: Identifier of the requested application to start streaming

IN: function pointer callback to inform when done, must NOT be NULL

OUT: struct with path to game build, metadata?

Return value

gfnSuccess: On success

Otherwise, appropriate error code

TitleExited

C	GfnRuntimeError **gfnTitleExited**(char* pchPlatformId, const char* pchPlatformAppId);

Description

Notifies GFN that an application has exited. GFN will then start the shutdown process for that application.

Note that this is for use by platform clients only and assumes the application has already completed execution. To shutdown from within an application itself, ShutdownGFNLinkSDK is used.

Usage

Use after an application has exited.

Parameters

pchPlatformId: Identifier of the launcher service, e.g. “Steam”

pchPlatformAppId: Identifier of the requested application to start streaming

Return value

gfnSuccess: On success

Otherwise, appropriate error code