Integrating IsOnline Services

Core Concepts

This guide uses the concepts of Client, IsOnline Notifications, IsOnline Session, IsOnline Status, and others, as they are described in “Core Concepts”.

Overview of IsOnline Lifecycle

This is the typical sequence of requests and responses that a client follows during its normal operation. With the exception of direct queries, all IsOnline events exist within the context of a single IsOnline session.

The standard intended lifecycle of an IsOnline service client goes like this:

1. Incepting an IsOnline Session
2. Subscribing to a Notification
3. Updating IsOnline
    - Receiving Notifications
    - Closing a Session

Phases 2 and 3 can overlap freely. For example, a client can add or remove subscriptions at any time with an active IsOnline session. They constitute the “live activity” of an IsOnline session.

Incepting an IsOnline Session

The following diagram illustrates the steps for incepting an IsOnline session.

[FLOWCHART OMITTED]

1. The Client calls or the IsOnline service responds with `MakeIsOnlineSession()`, with parameters containing the client information: platform, product, locale, and `UserID`. The `UserID` is passed in a JWT header.
2. The Client calls or the IsOnline service responds with `MakeIsOnlineSessionResponse`, containing the `SessionID` and the `SessionToken` for the newly created session. The client should keep those for later use.
3. The Client calls or the IsOnline service responds with `ConnectToIsOnlineSession()`, with parameters containing the `SessionToken` received above, as well as the initial IsOnline state for the client. This activates the IsOnline session and sets its IsOnline state. If there are clients that are subscribed to this user, a notification is sent to them about a new session going online. The response stream for this call is the IsOnline events stream.

During initialization, the client performs two RPC calls: `MakeIsOnlineSession()` followed by `ConnectToIsOnlineSession()`. The first call sets the client parameters and provides session authentication information for the client. In particular, the Session Token received in the response is needed in all subsequent operations (updating own IsOnline state, managing subscriptions). The second call opens the stream through which all IsOnline notifications are sent to the client.

Once both operations succeed, the client (the IsOnline session of the user on the particular client) is officially online as far as the IsOnline service is concerned, and other IsOnline sessions that have a subscription for the user would receive a notification.

Subscribing to Notifications

Once a client has a Session Token, it can start subscribing to the IsOnline updates of other users. A subscription request for a single target user, `SubscribeToUserIsOnline()`, does the following:

1. Retrieves the current IsOnline state of all sessions of the target user, and reports to the requesting client via the Events Stream (set up during the initialization phase).
2. Sets up the internal subscription for the future updates to all of the target user's sessions (new sessions, updates or closures of existing sessions).

Based on the online state of the target user, the immediate outcome of the subscription can be slightly different. See the following two cases:

Case 1 – Target user has one or more online sessions

The requesting client receives an IsOnline Notification for every IsOnline session on record.

[FLOWCHART OMITTED]

1. `SubscribeToUserIsOnline()` passes the session token and the target user ID. Server sets up the subscription for future updates.
2. The async request is sent to the internal storage for all sessions of the target user.
3. `SubscribeToUserIsOnlineResponse` indicates that subscription has been set up.
4. A result arrives from the storage component.
5. The IsOnline service transforms the result into a `IsOnlineNotification` and sent through the events stream. The event stream was set up by `ConnectToIsOnlineSession()` during initialization.
6. 4 and 5 repeat for every active session of the target user, and so on…

Case 2 – User is offline (no active sessions)

When the target user has no active sessions, the IsOnline Service sends a notification that contains no session information, the user-level flag which is set to `offline`, and the "last seen online" timestamp.

[FLOWCHART OMITTED]

1. `SubscribeToUserIsOnline()` passes the session token and the target user ID.
2. Server sets up the subscription for future updates. The async request is sent to the internal storage for all sessions of the target user.
3. `SubscribeToUserIsOnlineResponse` indicates that the subscription has been set up.
4. Storage indicates that there are no active sessions.
5. Storage responds with a record of user-level information. In practice, this step is done in both cases, but its response only matters for the fully offline case.
6. Response: user-level information including the last seen online timestamp.
7. A `IsOnlineNotification` is generated and sent through the events stream. The event stream was set up by `ConnectToIsOnlineSession()` during initialization. The notification contains an offline user-level flag and the "last seen online" timestamp.

Friend List Subscription

Keeping track of all friends' IsOnline is a frequent use case, and the IsOnline service has a dedicated RPC for that: `SubscribeToFriendsIsOnline()`. The single parameter for the call is the session token. The call does the following:

1. Retrieves the current friend list for the user.
2. Similar to the single-target subscription, retrieves the current state for every user on the friends list, and communicates it via Notifications on the event stream.
3. Similar to the single-target subscription, sets up internal subscriptions for all users on the friends list.
4. Starts listening for the updates from Friends Notifications, and updating the subscriptions accordingly.

The output for the `SubscribeToFriendsIsOnline()` call is similar to the single-target call: a simple acknowledgement for the call itself, the at least one IsOnline Notification on the events stream for every user on the friends list. In effect, the set of notifications would be the same if each friend is subscribed to individually (IsOnline-update-related race conditions notwithstanding).

Updating IsOnline

The IsOnline session status is set in two ways: at the session initialization, and through live updates. The IsOnline updates can be sent at any time while an IsOnline session is active (that is, has an active events stream created by `ConnectToIsOnlineSession()` during the initialization phase).

The IsOnline status can be updated via the `UpdateIsOnlineSession()` RPC. The parameters for the call are the session token and the `IsOnlineUpdate` payload. The call has the following effects:

1. The update payload is stored as the "current state" of the session, to be later retrieved as part of subscription or a direct get-IsOnline request.
2. The update is broadcast to all of the subscribers of the session's user. Before being sent to each subscribing client, the update would be transformed into an IsOnline Notification and localized.

[FLOWCHART OMITTED]

1. `UpdateIsOnlineSession()` call, supplies a session token and the update data. The session token is decrypted to get a `SessionID`.
2. Update data stored as the latest known IsOnline state of the session.
3. Update is broadcast to all subscribers of the session's user.
4. Update is acknowledged by the user.