**API Reference for Direct Line Client’s StreamingConversations**

This guide describes the new APIs that are used to interact with the Direct Line App Service Extension preview.

***DirectLineClient***

DirectLineClient exposes one new property, ‘StreamingConversations’, where the new functionality is located:

public partial class DirectLineClient

{

public virtual IStreamingConversations StreamingConversations { get; private set; }

}

***IStreamingConversations***

The IStreamingConversations interface has all of the methods needed to connect the WebSocket to Direct Line App Service Extension, to manage that connection, and to send all of the familiar commands to Direct Line through this WebSocket. To use the methods on this interface, you must first call ConnectAsync. The interface looks like:

public partial interface IStreamingConversations

{

/// <summary>

/// A property to determine if the StreamingConversations is connected or not

/// </summary>

bool IsConnected { get; }

/// <summary>

/// An event that is signalled whenever the StreamingConversations

/// becomes disconnected

/// </summary>

event DisconnectedEventHandler Disconnected;

/// <summary>

/// Connects the StreamingConversations class to a service using the

/// DirectLineClient's endpoint and credential value

/// This method takes a callback, receiveActivities, which is invoked whenever

/// DirectLine needs to send Activities from the bot back to the client

/// </summary>

/// <param name="conversationId">The conversationId that will be

/// used for this connection.

/// </param>

/// <param name="receiveActivities">A callback that is invoked whenever

/// Activities are sent back to the client

/// </param>

/// <param name="cancellationToken">Cancellation Token</param>

/// <returns></returns>

Task ConnectAsync(

string conversationId,

Action<ActivitySet> receiveActivities,

CancellationToken cancellationToken = default(CancellationToken));

/// <summary>

/// Disconnects the StreamingConversations class from the service

/// </summary>

void Disconnect();

/// <summary>

/// Starts a conversation with the bot. This method will signal to the Direct Line

/// instance that user's who are added to the conversation will trigger

/// ConversationUpdate Activities to be sent to the bot

/// </summary>

/// <param name="tokenParameters">Optional configuration parameters about

/// the conversation</param>

/// <param name="cancellationToken">Cancellation Token</param>

/// <returns></returns>

Task<Conversation> StartConversationAsync(

TokenParameters tokenParameters = null,

CancellationToken cancellationToken = default(CancellationToken));

/// <summary>

/// Reconnects a conversation with the bot. After calling this method,

/// users who are added to the conversation

/// will NOT cause ConversationUpdate messages to be sent to the bot

/// </summary>

/// <param name="conversationId">The conversationId to reconnect to.</param>

/// <param name="watermark">Activity watermark</param>

/// <param name="cancellationToken">Cancellation Token</param>

/// <returns></returns>

Task<Conversation> ReconnectToConversationAsync(

string conversationId,

string watermark = null,

CancellationToken cancellationToken = default(CancellationToken));

/// <summary>

/// Sends an UpdateActivity to the bot

/// </summary>

/// <param name="conversationId">The conversation id</param>

/// <param name="activityId">The id of the activity to update</param>

/// <param name="activity">The updated activity</param>

/// <param name="cancellationToken">Cancellation Token</param>

/// <returns></returns>

Task<ResourceResponse> UpdateActivityAsync(

string conversationId,

string activityId,

Activity activity,

CancellationToken cancellationToken = default(CancellationToken));

/// <summary>

/// Sends an activity to the bot

/// </summary>

/// <param name="conversationId">The conversationId</param>

/// <param name="activity">The activity to send</param>

/// <param name="cancellationToken">Cancellation Token</param>

/// <param name="attachmentStreams">An optional set of streams (attachments)

/// that should accompany the Activity</param>

/// <returns></returns>

Task<ResourceResponse> PostActivityAsync(

string conversationId,

Activity activity,

CancellationToken cancellationToken = default(CancellationToken),

params AttachmentStream[] attachmentStreams);

/// <summary>

/// Sends a set of streaming attachments to the bot

/// </summary>

/// <param name="conversationId">The conversationId</param>

/// <param name="userId">The userId associated with sending these attachments</param>

/// <param name="cancellationToken">Cancellation Token</param>

/// <param name="attachmentStreams">A set of streams (attachments) that

/// should accompany the Activity</param>

/// <returns></returns>

Task<ResourceResponse> UploadAttachmentsAsync(

string conversationId,

string userId,

CancellationToken cancellationToken = default(CancellationToken),

params AttachmentStream[] attachmentStreams);

}

**AttachmentStream**

The AttachmentStream class encapsulates a stream that you want to send from the client to the bot.

public class AttachmentStream

{

public string ContentType { get; }

public Stream ContentStream { get; }

}