Mock Channel Implementation for Performance Testing of BOT

Table of Contents

[Document Overview 4](#_Toc7527804)

[Implementation Approach 4](#_Toc7527805)

Document Overview

The document briefs on

* Why a mock channel is to be used for performance testing a BOT.
* Solution of Mock Channel Implementation.
* Parameters to be supplied while publishing the Mock Channel App Service to Azure.

Implementation Approach

Directline approach to send requests to BOT on load is not recommended for performance testing.

Throttling occurs with Directline especially for single (or limited) IP scenarios.

To load test a BOT, we implement a mock channel which is a call back service to which the BOT framework sends the activity message (response message) while responding to a BOT request received from user.

* When a request is sent to BOT( say from client: Visual Studio Webtest), response of BOT is sent back to BOT Connector Service by default. To capture the timestamp of the BOT response ( response body has a timestamp), we create a mock channel to which this response is redirected.
* Within this mock channel, methods for testing BOT are defined.
  + When BOT’s endpoint receives a request (message with JSON Serialized body containing the ServiceUrl which is the mock channel end point) from user, using the information in that request, an [Activity](https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-connector-api-reference?view=azure-bot-service-4.0#activity-object) object is created for response to user request.
  + To reply to a specific message within the conversation, **ReplyToActivity** method is used that is implemented in the mock channel.
  + Within this method, BOT Response is captured and saved to Storage Table that can be retrieved later. The time stamp of response received is also captured in the Storage table for that activity.
  + **StoreResponse** method implemented saves the response to Storage Table.
  + To retrieve the response from storage table, request( from client) to be made to **GetConversation** method (with the same conversation ID and Activity ID sent earlier in BOT request within the JSON Serialized body).

**Inputs to be supplied in the solution:**

Storage Account details of the Azure Storage Table: AzureStorageAccount, AzureStorageSecret

Storage Table Name to save the response: TableName

