## Simple pass through implementation via MPGW

|  |
| --- |
| 🡪Once, we have a domain in place we can create a configurable service in the MPGW.  🡪The MPGW is versatile and powerful object in the DP.  🡪It has various capabilities.  🡪Primary task of this object is to convert one protocol to another.  🡪Eg. We have a requirement that our back end systems be on pure http, however, the clients should access via HTTPS.  🡪In such cases, we can use this object to configure the DP in such a way that DP acts as SSL termination end point.  🡪If we are using RESTful services, then MPGW is best object to use, instead of Webservice Proxy.  🡪The MPGW also has other uses like it can host SFTP gateway for the organization.  🡪Following use case demonstrates a simple implementation of MPGW for a REST service running on Apache Tomcat. |
| 🡪Consider a REST service running on Apache Tomcat on a machine.  🡪When the service is accessed, via browser or a client (Eg. NetTool), the service will send a JSON response.  🡪The working URL while writing these notes as below:  http://192.168.1.159:9000/APIMgmtRestFulServiceLengthFinder/testrest/lengthFinder/sreedhar  🡪Incase the URL is down or not working, we have the Project also embedded in this folder that can be deployed to a Tomcat server.  🡪Little explanation about the URL is as below:  🡪There is a web application which has context root as : APIMgmtRestFulServiceLengthFinder.  🡪This application was developed in java. Note that it can be developed in any technology.  🡪Same is hosted on Tomcat7 on machine 192.168.1.159 port 9000.  🡪The application has a REST URI as below, which is appended after the Context Root.  testrest/lengthFinder/sreedhar  🡪 /testrest is the URL prefix that is provided in the web.xml of the webapplication.  🡪/lengthfinder is the actual REST URI that is mentioned on the JAVA Class.  🡪/sreedhar is the parameter that is being passed.  🡪When this URI accessed, the back end service running on Tomcat will give us the length of the parameter. Eg. sreedhar would give us 8.  🡪Note carefully the IP address and port number, as we will not be using the same while accessing via Datapower.  🡪Now, given this scenario, if we want our back end service like above to be secured and to be accessed via DMZ based application like Datapower, we need to follow following steps to configure it. |

|  |
| --- |
| Following sequence of activities will enable our back end services to be secure via Datapower.  🡪Note that this is the simplest implementation where both sides are on http protocol and datapower will simply allow a passthrough for request and response.  🡪Logon to the Datapower and select the workspace (domain) that we have created.  🡪Select 'Multi Protocol Gateway' option from services.  🡪Click on the 'ADD' button to get following view after filling necessary fields. |
| 🡪The explanation for the circled parts is as below.  🡪1. This is the logical name for the MPGW configuration that has been given.  🡪2. This is the back end URL where our REST Services are running.  🡪Note that this is a static back end URL that has been provided by selecting the static-backend radio button in the Type.  🡪Here, we have not mentioned the full URL of the back end service.  🡪We have configured, in the POLICY (Option 3), that let any pattern come after this URL, datapower will accept and route it to back end service. How it is done check Option 3.  🡪 |
| Matching Rule window |
| 4. Configuring Front side handler |