# ENTERING
# THE XML API

ATPI

For use within the ICT department of ATPI

## TABLE OF CONTENTS

# 1. INTRODUCTION

This document describes the entry point for our XML API. Our XML API is based on SOAP 1.1 and has one main entry point namely ITicketWindowSOAPAPI. The reader of this document should have expert knowledge of executing SOAP methods via the HTTP transport protocol. It is possible to get the data from the API in GZIP format by adding the "accept-encoding=GZIP" into the HTTP header.

The next chapters will describe the entry point and what methods are available within this entry point.

## 2. THE MAIN ENTRY POINT OF THE ATPI XML API

The XML API uses the /soap as the main entry point. For example the url to the soap API would be "http://server/tc.dll/soap".

The WSDL for our webservice is being published on the following entry point: "http://server/tc.dll/wsdl/ITicketWindowSOAPApi"

Next to the soap entry point there needs to be a SOAP action defined. The following methods are available from within our SOAP API

1. HelloEngine – this is a simple way of testing our XML API. When requesting this method you can provide a text to be send back. This call returns the following: Hi there xxx, this is server: SERVER telling you it is Friday, 29 July 2011 and the time is 09:20:24,20. Your IP Address is: 127.0.0.1
2. ExecuteXML – Our main method to use when querying our system. When asking for flights, bookings, signons etc. it will all go through this main method.

The methods are provided via a soap action header. Below a C++ sample of how this header looks like.

```
private void sendRequest(char* URI, std::string request)
{
        myHTTPObject = new HTTPObject();
        myHTTPObject->SetRequestHeader("SOAPAction",
                                        "urn:TC_SOAP_API-ITicketWindowSOAPApi#ExecuteXML");
        myHTTPObject->SetRequestHeader("Content-Length", request.length());
        myHTTPObject->SetRequestHeader("Content-Type", "text/xml; charset=utf-8");

        try
        {
                myHTTPObject->Post(URI, request.c_str(), request.length());
                ….
        } catch (exception e)
                …
        }
}

public void foo()
{
        sendRequest("http://server/tc.dll/soap", "<TWSRequest><action>LOGON</action></TWSRequest>");
        …
}
```

## 3. THE HELLOENGINE METHOD

This method is purely for testing the communication with our SOAP API. It can allow an additional string value which is being sent back in the response. When executing this request the SOAP API returns the server name in combination with the date and time as an answer.

Below a sample of how to execute such request in c++

```
private void sendRequest(char* URI, std::string data)
{
        myHTTPObject = new HTTPObject();
        myHTTPObject->SetRequestHeader(“SOAPAction”,
                                         “urn:TC_SOAP_API-ITicketWindowSOAPApi#HelloEngine”);
        myHTTPObject->SetRequestHeader(“Content-Length”, data.length());
        myHTTPObject->SetRequestHeader(“Content-Type”, “text/xml; charset=utf-8”);

        try
        {
                myHTTPObject->Post(URI, data.c_str(), data.length());
                ….
        } catch (exception e)
                …
        }
}

public void foo()
{
        sendRequest(“http://server/tc.dll/soap”, “John doe”);
        …
}
```

## 4. THE EXECUTEXML METHOD

This is the main method used for querying our XML API. With this method you can login to our system, book flights, retrieving bookings etc.

So basically all of our XML requests are being sent over this execution method.

Below a sample of how this can be done in C++

```
private void sendRequest(char* URI, std::string request)
{
        myHTTPObject = new HTTPObject();
        myHTTPObject->SetRequestHeader(“SOAPAction”,
                                     “urn:TC_SOAP_API-ITicketWindowSOAPApi#ExecuteXML”);
        myHTTPObject->SetRequestHeader(“Content-Length”, request.length());
        myHTTPObject->SetRequestHeader(“Content-Type”, “text/xml; charset=utf-8”);

        try
        {
                myHTTPObject->Post(URI, request.c_str(), request.length());
                ….
        } catch (exception e)
                …
        }
}

public void foo()
{
        std::string request =
“<TWSRequest><action>LOGON</action><subaction>MEMBER</subaction>….</TWSRequest>”;

        sendRequest(“http://server/tc.dll/soap”, request);
        …
}
```

Responses are being sent back in XML and will always start with a <TWSResponse> token.