Mobile Integration of CICS® Transaction Gateway and CICS Transaction Server for z/VSE®

Michael Blum
blumm@de.ibm.com

Table of Contents

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

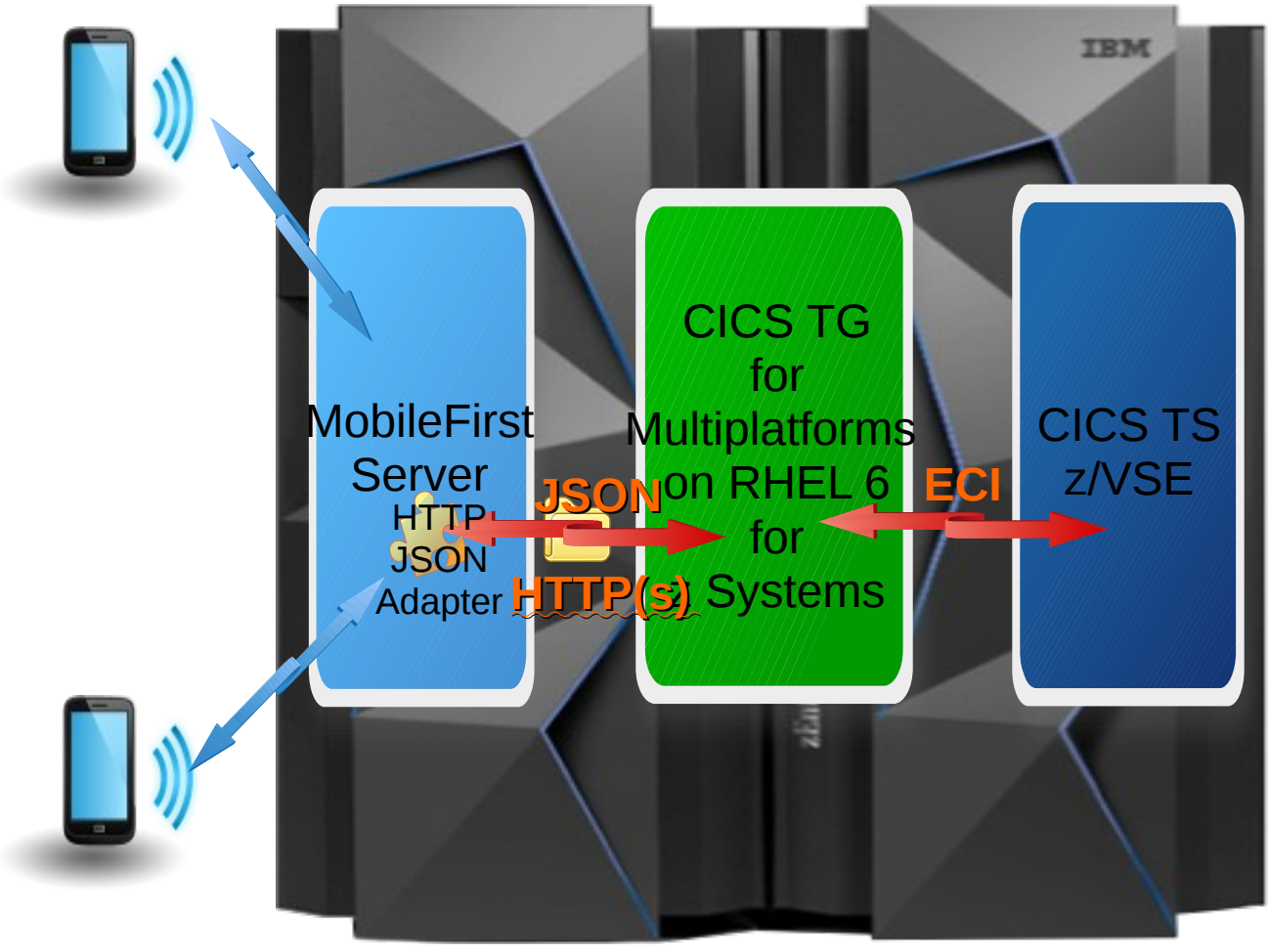
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

1 Introduction

Many applications are run as transactions on CICS® TS for z/VSE® . They might have been written in COBOL,PL/I or C language and are of significant value for the business of their companies. CICS Transaction Gateway (CTG) is often used to access CICS transactions from a web server or remote platform.
Today Mobile Apps are getting increasingly popular and it can be essential to connect them to those CICS TS applications to exploit the value to new users. IBM MobileFirst™ V7 and CICS TG V9 are the means to get this done in a quite efficient manner.
The IBM MobileFirst Platform provides the components to mobilize your CICS Transaction. It uses server-side applications and Adapters to communicate with back-end services, such as CICS applications. Adapters connect to enterprise applications, deliver data to and from mobile applications, and perform any necessary server-side logic on sent data.
This paper shows how easily this can be done. It focuses on a MobileFirst HTTP(S) adapter that posts a HTTP request with a JSON payload to the CTG. The CTG is set up to serve the requests to the CICS TS at the z/VSE enterprise server as backend.

1.1 Mobile Multi-Tier Client/Server Toplogy

The end-to-end topology assumes a client mobile device such as Android, IOS, Windows, or Blackberry phones. It uses a MobileFirst server as middle-tier server and a



CICS TG for Multiplatforms and finally the back-end server which is CICS TS for z/VSE. The focus is on MobileFirst adapter development and configuration of the middle-tier software and the CICS backend server connectivity.

The example in this workbook is implemented based on:

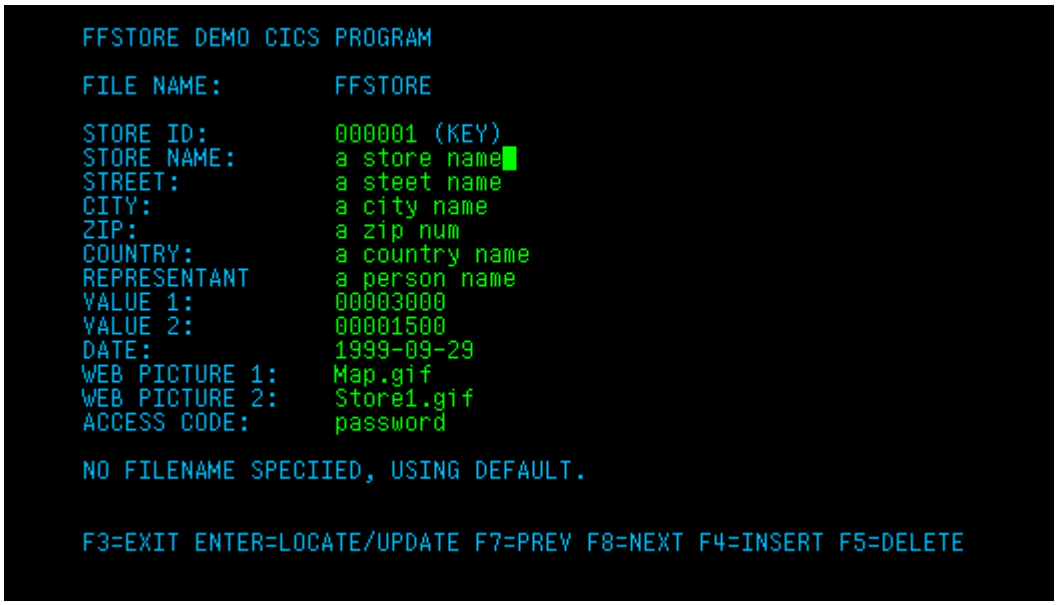
- IBM MobileFirst Studio 7.0 (MobileFirst Plugin used with Eclipse Luna SR2(4.2.2))
  - Installed on RHEL 6.6 X86 64
- IBM CICS Transaction Gateway for Multiplatforms Version 9 Release 1
  - installed on RHEL 6.6 for Linux on z Systems™
- IBM CICS TS for z/VSE V2.1
  - installed on z/VSE V6.1

1.2 CICS Transaction - FFStore Demo

z/VSE demo-application Fictitious Food Store (FFStore), a simple CICS application is used as an enterprise application sample here. FFStore uses VSAM to store and retrieve data about stores on VSAM record basis. FSTIO-MAP is the COBOL copybook for language structure that describes the interface for the CICS transaction FFST:

```
03 FSTIO-MAP.
   05 ACTION              PIC 9(8) COMP.
   05 RETURN-CODE         PIC 9(8) COMP.
   05 FILE-NAME           PIC X(8).
   05 STORE-ID            PIC X(6).
   05 STORE-NAME          PIC X(25).
   05 LOC-STREET          PIC X(25).
   05 LOC-CITY            PIC X(25).
   05 LOC-ZIP             PIC X(10).
   05 LOC-COUNTRY         PIC X(25).
   05 LOC-REP             PIC X(20).
   05 VAL1                PIC 9(8) COMP.
   05 VAL2                PIC 9(8) COMP.
   05 DATE                PIC X(10).
   05 WEB-PIC1            PIC X(20).
   05 WEB-PIC2            PIC X(20).
   05 A-CODE              PIC X(10).
   05 FILLER              PIC X(6).
```

Calling CICS transaction FFST with FFSTORE as VSAM file name from a z/VSE terminal session shows up with a panel and data as well as PF keys for certain actions .



2 Mobile Integrated CICS TS Application

The CICS TG will be used to expose a HTTP/JSON web service for the traditional COBOL/COMMAREA application FFSTORE which runs in CICS TS for z/VSE server. JSON (JavaScript Object Notation) is a lightweight data-interchange format often used for mobile payloads on top of HTTP. The JSON payloads are automatically converted to a COMMAREA payload and vice versa that is passed using CTG's External Call Interface (ECI) to and from the FFStore application.
IBM MobileFirst Development Studio is used to develop an HTTP/JSON type adapter. The integrated MobileFirst Server is used for the deployment/test of the adapter. The adapter posts HTTP/JSON request to the CICS TG that functions as HTTP to ECI converter. The adapter will receive and handle the JSON response.

2.1 CICS TG for Multiplatforms V9.1 on RHEL 6.6 for z Systems.

The installation was performed according to the information in the CICS CTG Knowledge Center. The CICS CTG Knowledge Center provides also more details, which are not outlined in here.

2.1.1 Generation of a JSON Schema and WSBind file from COBOL Language

The so called bottom-up approach is used when having a data structure written in C, COBOL, or PL/I that describes how a certain CICS program expects the data to be arranged. The bind file generation process produces also JSON schemas that can be used to write applications that invoke the web service.
A WSBind file describes, which CICS program to call when a JSON web service is invoked and how JSON is converted to a channel or COMMAREA payload.

The **ctgassist** command is used to generate a web service binding file and the JSON schema files that are used with web service applications. The input to ctgassist is a parameter file, which is **ffst-ctgls2js-cobol.txt** (below) for the FFST CICS application.

The syntax is: ctgassist [-X] <parameter file>
Optional -X switches on tracing of the assistant. It produces useful information for correction of the parameter file content.

The command ‘ctgassist ffst-ctgls2js-cobol.txt’ is used to generate the two JSON schema files for request and response:
`ffst-cobol-request.json`
`ffst-cobol-response.json`
`and the` web service binding file:
`ffst-cobol.wsbind`
Those files are referenced during the configuration of the CICS TG.

2.1.1.1 File: ffst-ctgls2js-cobol.txt

The file was derived from the sample CTGLS2JS.txt located in /opt/ibm/cicstg/samples/webservices. Lines beginning with '#' are comments and provide original information/instructions. Below is the file content as used for FFStore. The changes are the lines in blue.

```
#
# See the product documentation for the full list of parameters.
#
# LANG=<COBOL|C|CPP|PLI-Enterprise|PLI-Other>
LANG=COBOL
MAPPING-MODE=LS2JS
# JSON-SCHEMA-REQUEST=<Request schema file>
JSON-SCHEMA-REQUEST=/root/mobile-ctg/ffst/ffst-cobol-request.json
# JSON-SCHEMA-RESPONSE=<Response schema file>
JSON-SCHEMA-RESPONSE=/root/mobile-ctg/ffst/ffst-cobol-response.json
# LS-REQUEST=<Request language structure file>
LS-REQUEST=/root/mobile-ctg/ffst/ffst-request.cb
# LS-RESPONSE=<Response language structure file>
LS-RESPONSE=/root/mobile-ctg/ffst/ffst-response.cb
# WSBIND=<WSBind file>
WSBIND=/root/mobile-ctg/ffst/ffst-cobol.wsbind
# PGMNAME=<Program name>
PGMNAME=FFSTIO
# PGMINT=<CHANNEL|COMMAREA>
PGMINT=COMMAREA
# TARGET-CICS-PLATFORM=<zOS|AIX|HP-UX|Solaris|IBM-i|VSE|LinuxI|Windows>
TARGET-CICS-PLATFORM=VSE
# <Other options>
```

2.1.2 CICS TG - Basic Configuration

The configuration file of the CICS TG is ctg.ini. It's default location is /opt/ibm/cicstg/bin/. The structure/sections of ctg.ini and the parameter are explained in detail here: Configuration parameter reference.

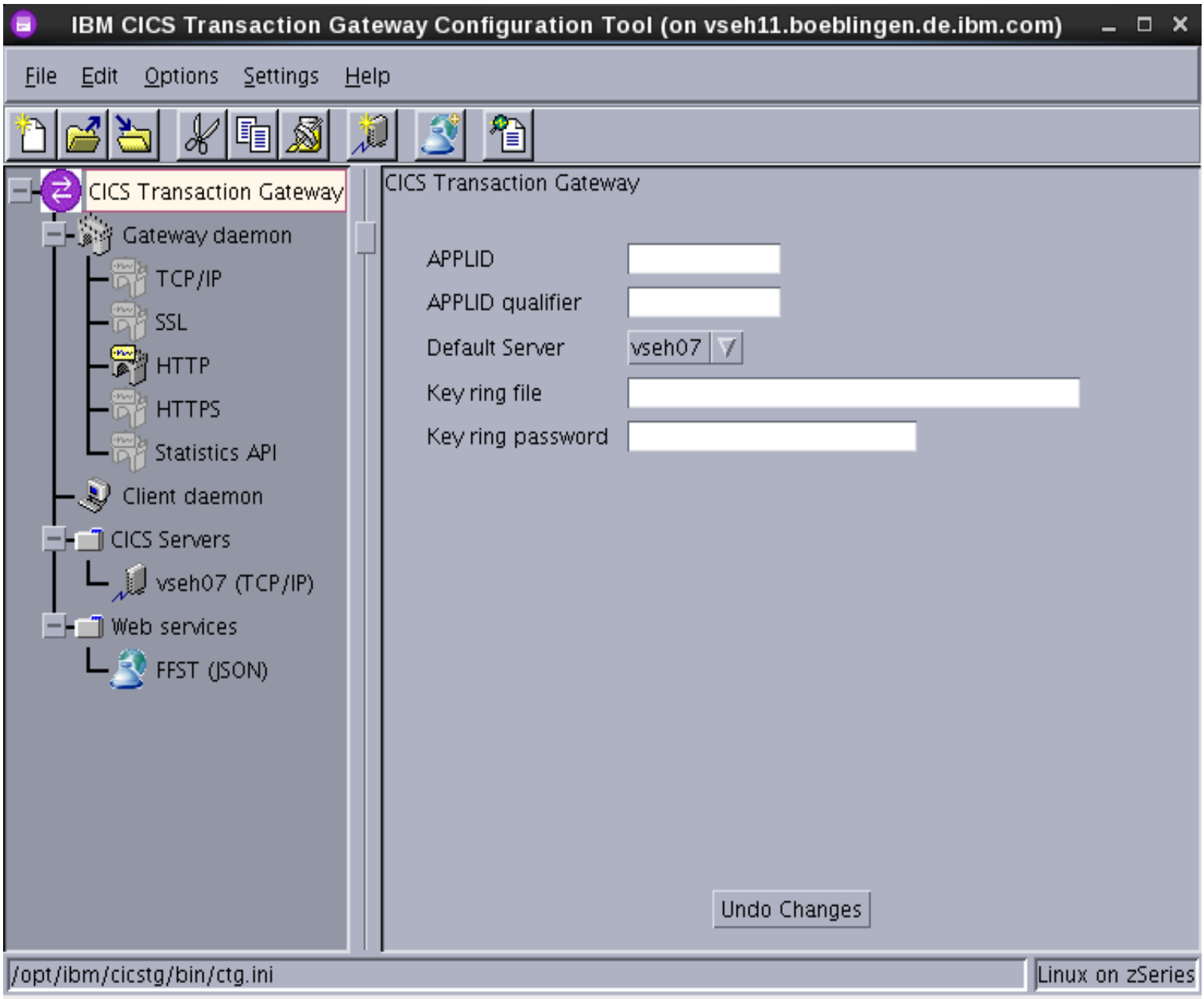
The CICS TG Configuration Tool can help to easily navigate through the many configuration settings, which are important in the scope of the FFStore demo scenario here. The tool uses the X Window System. It is started on the command line by entering **ctgcfg**. More information about this tool can be found here: Creating a configuration file.

ctgcfg updates the ctg.ini file.

The next screenshots of the CICS TG configuration tool reflect the explicit (non-default) settings done for the FFStore demo connection. Some values are further explained for your convenience. Use the reference documentation in the Knowledge Center for more details.

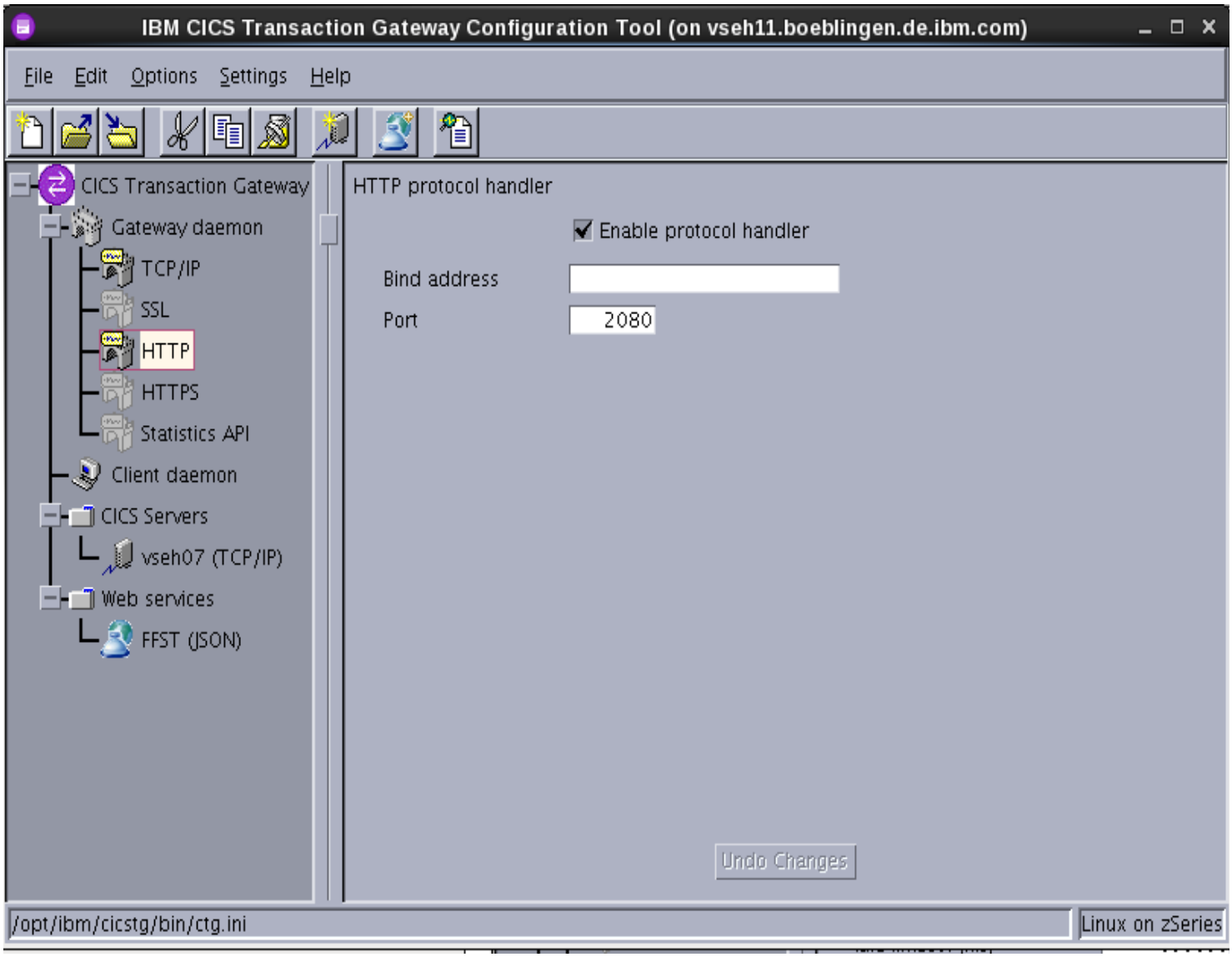
2.1.2.1 CICS Transaction Gateway

- Default Server is set to the hostname of the z/VSE system with the CICS TS.



2.1.2.2 HTTP Protocol Handler

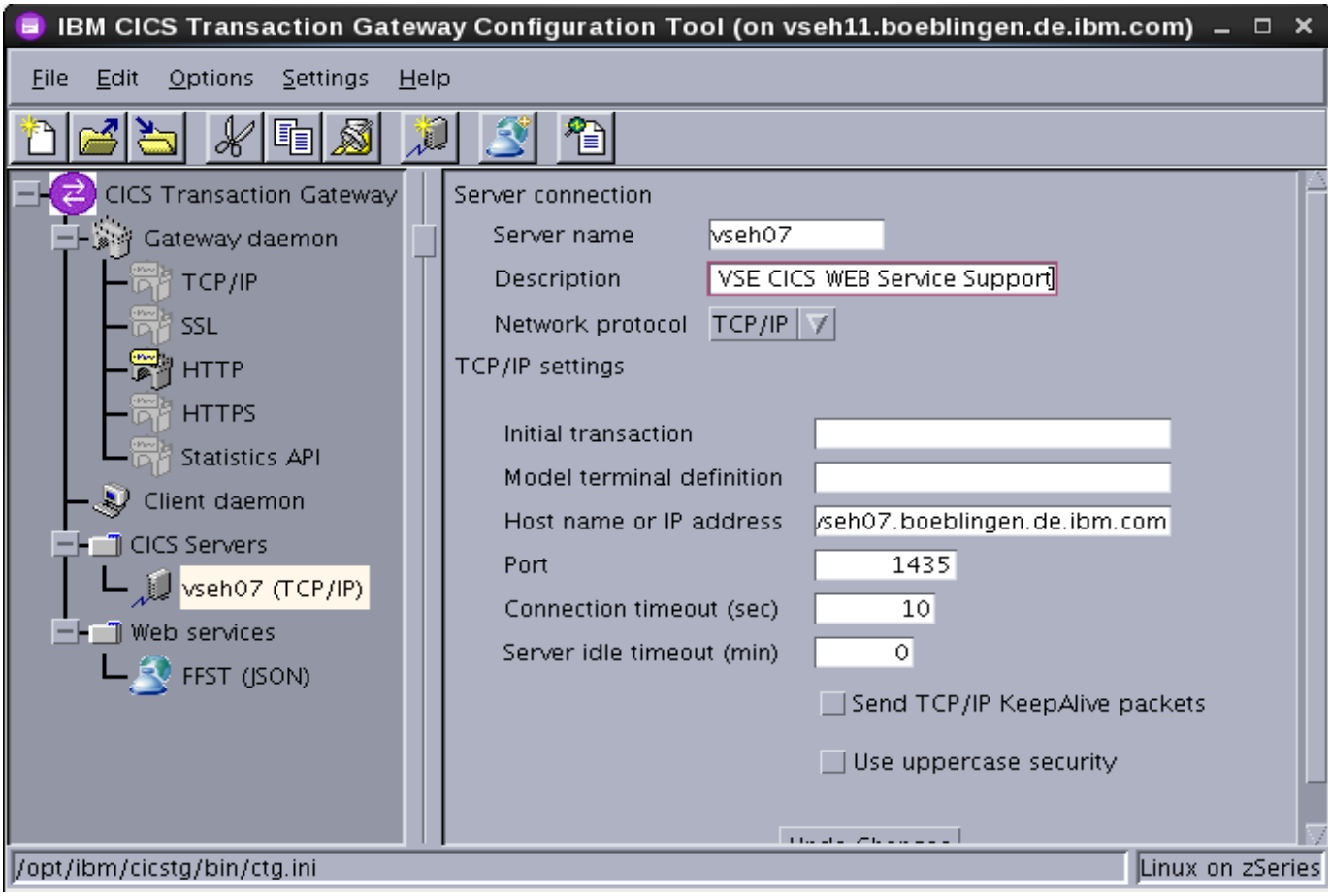
- Bind address: An explicit ip-address or host name that the HTTP protocol handler is bound to. Can be used when multiple TCP/IP adapters are configured/installed in the system.
- Port: The HTTP protocol handler listens on the HTTP port given (here - 2080) for incoming requests. This is the target port as to be referenced in a IBM MobileFirst Adapter implementation (e.g. HTTP_JSON_CTG.xml).



2.1.2.3 CICS server definition

Configuring a TCP/IP CICS Server definition. This panel shows a connection to a CICS TS running in z/VSE server. A connection to a z/VSE CICS TS server and ECI service is configured.

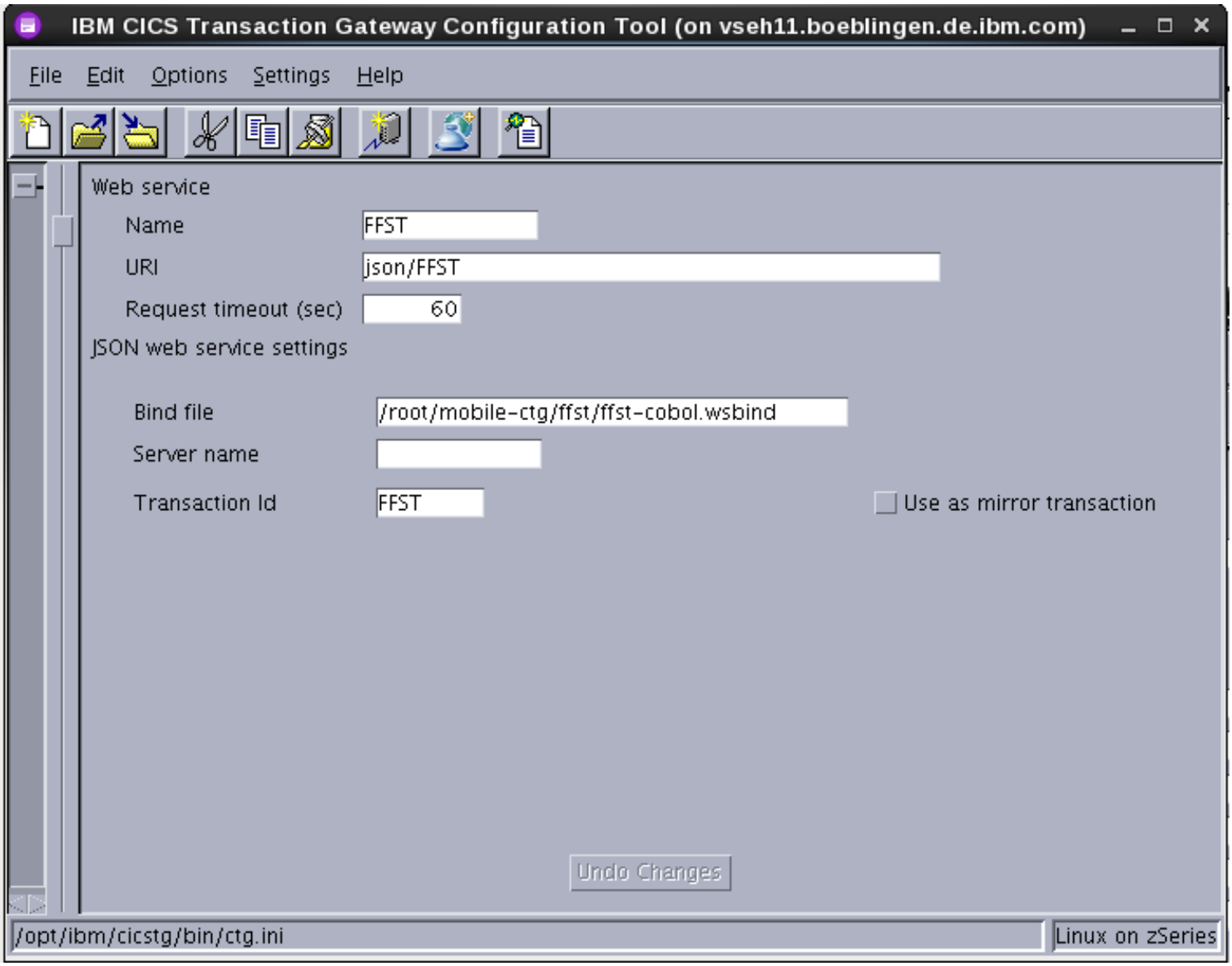
- Server name: vseh07 defines a logical CICS server name used to reference the actual CICS TS server. The logical CICS server name is used in API requests.
- Host name of IP address: vseh07.boeblingen.de.ibm.com is the hostname as known in the TCP/IP network.
- Port: E.g.1435 is the port of an ECI service as configured in z/VSE CICS TS. This can be inquired by 'CEMT I TCPIPSERVICE(ECI)'.



2.1.2.4 Web service

A JSON web service is configured on the following panel.

- Name: FFST a web service name
- URI: json/FFST that is a unique uri used by the web service.
  Bind file: The location of the WSBind file. It basicallydecribes, which CICS program to call when the web service is invoked and how JSON is converted to a channel or COMMAREA payload.
- Transaction Id: FFST, which is the transaction identifier passed to the CICS program.



2.1.3 The CICS® Transaction Gateway configuration file - ctg.ini

The main CTG configuration parameter settings used for the CICS FFStore example-transaction are in blue. The meaning of those parameter is described in the chapters for the panels.

```
SECTION PRODUCT
    DEFAULTSERVER=vseh07
ENDSECTION

SECTION GATEWAY
    closetimeout=10000
    connectionlogging=off
    cicslogging=off
    initconnect=1
    initworker=1
    maxconnect=100
    maxworker=100
    maxhttpconnect=100
    noinput=off
    dnsnames=off
    tfile=/root/mobile-ctg/http.trace
    trace=on
    workertimeout=10000
    adminport=2810
    statint=030000
    stateod=000000
    statsrecording=off

    SUBSECTION HTTP
        Port=2080
        Bind=
    ENDSUBSECTION

    log@info.dest=console

    log@error.dest=console
ENDSECTION

SECTION CLIENT = *
    TERMINSTLOGGING=N
    LOGFILE=cicscli.log
    LOGFILEINFO=cicscli.log
    MAXBUFFERSIZE=32
    MAXREQUESTS=256
    MAXSERVERS=10
    MAXWRAPSIZE=0
    SRVRETRYINTERVAL=60
    TERMINALEXIT=EXIT
    TRACE=TRN,DRV.2,CCL
    TRACEFILE=cicscli.bin
ENDSECTION
```



```
SECTION WEBSERVICE = FFST
    URI=json/FFST
    TIMEOUT=60
    BINDFILE=/root/mobile-ctg/ffst/ffst-cobol.wsbind
    TRANSACTIONID=FFST
    DEFAULTMIRROR=Y
ENDSECTION

SECTION SERVER = vseh07
    DESCRIPTION=Mobile VSE CICS WEB Service Support
    SRVIDLETIMEOUT=0
    UPPERCASESECURITY=N
    PROTOCOL=TCPIP
    NETNAME=vseh07.boeblingen.de.ibm.com
    PORT=1435
    CONNECTTIMEOUT=10
    TCPKEEPALIVE=N
ENDSECTION

SECTION DRIVER = TCPIP
    DRIVERNAME=CCLIBMIP
ENDSECTION
```

2.1.4 A Simple Security Connection Setup - CICS TG to CICS TS

For the sake of a demo only scenario the **cicscli** tool is used here to define a simple authentication scheme (user ID password) which the CICS CTG provides when connecting CICS TS on z/VSE. The actual User ID and password can be set by the cicscli command, which is used to administer the CICS TG Client daemon.

The syntax used is

cicscli -c=<servername> -u=<userid> -p=<password>

The command details of this format is shown here: Setting security for server connections.

2.2 IBM MobileFirst

2.2.1 A Rest – HTTP Adapter

The IBM MobileFirst Adapter is of type HTTP(S). It's request/response payload uses JSON. In Chapter *2.1.1 Generation of a JSON Schema and WSBind file from COBOL Language* is described how to use the **ctgassist** tool to generate the JSON schemas for JSON payloads, one for the requests and another one for responses. The schema files generated are:
`ffst-cobol-request.json`
`ffst-cobol-response.json`

2.2.1.1 The JSON Request Schema - ffst-cobol-request.json

The following is an example JSON Request Schema generated by the ctgassist tool from the content of language structure data referenced in the example parameter file ffst-ctgls2js-cobol.txt. The language data source is decribed in 1.2 CICS Transaction - FFStore Demo. The schema describes in detail the rules how a JSON request payload is to be generated.

```
{
   "$schema":"http:\/\/json-schema.org\/draft-04\/schema#",
   "description":"Request schema for the FFSTIO JSON interface",
   "type":"object",
   "properties":{
      "FFSTIOOperation":{
         "type":"object",
         "properties":{
            "fstio_map":{
               "type":"object",
               "properties":{
                  "action":{
                     "type":"integer",
                     "maximum":4294967295,
                     "minimum":0
                  },
                  "file_name":{
                     "type":"string",
                     "maxLength":8
                  },
                  "store_id":{
                     "type":"string",
                     "maxLength":6
                  },
                  "store_name":{
                     "type":"string",
                     "maxLength":25
                  },
                  "loc_street":{
                     "type":"string",
                     "maxLength":25
                  },
                  "loc_city":{
```



```
            "type":"string",
            "maxLength":25
         },
         "loc_zip":{
            "type":"string",
            "maxLength":10
         },
         "loc_country":{
            "type":"string",
            "maxLength":25
         },
         "loc_rep":{
            "type":"string",
            "maxLength":20
         },
         "val1":{
            "type":"integer",
            "maximum":4294967295,
            "minimum":0
         },
         "val2":{
            "type":"integer",
            "maximum":4294967295,
            "minimum":0
         },
         "web_pic1":{
            "type":"string",
            "maxLength":20
         },
         "web_pic2":{
            "type":"string",
            "maxLength":20
         },
         "a_code":{
            "type":"string",
            "maxLength":10
         }
      },
      "required":[
         "action",
         "file_name",
         "store_id",
         "store_name",
         "loc_street",
         "loc_city",
         "loc_zip",
         "loc_country",
         "loc_rep",
         "val1",
         "val2",
         "web_pic1",
         "web_pic2",
         "a_code"
      ]
   }
```



```
         },
         "required":[
            "fstio_map"
         ]
      }
   },
   "required":[
      "FFSTIOOperation"
   ]
}
```

2.2.1.2 The JSON Response Schema – ffst-cobol-response.json

The following is an example JSON Request schema generated by the ctgassist tool from the content of language structure data referenced in the example parameter file ffst-ctgls2js-cobol.txt. The language data source is decribed in 1.2 CICS Transaction - FFStore Demo. The schema describes in detail the rules how a JSON response payload is to be generated.

```
{
   "$schema":"http:\/\/json-schema.org\/draft-04\/schema#",
   "description":"Response schema for the FFSTIO JSON interface",
   "type":"object",
   "properties":{
      "FFSTIOOperationResponse":{
         "type":"object",
         "properties":{
            "fstio_map":{
               "type":"object",
               "properties":{
                  "action":{
                     "type":"integer",
                     "maximum":4294967295,
                     "minimum":0
                  },
                  "file_name":{
                     "type":"string",
                     "maxLength":8
                  },
                  "store_id":{
                     "type":"string",
                     "maxLength":6
                  },
                  "store_name":{
                     "type":"string",
                     "maxLength":25
                  },
                  "loc_street":{
                     "type":"string",
                     "maxLength":25
                  },
                  "loc_city":{
```

```
            "type":"string",
            "maxLength":25
         },
         "loc_zip":{
            "type":"string",
            "maxLength":10
         },
         "loc_country":{
            "type":"string",
            "maxLength":25
         },
         "loc_rep":{
            "type":"string",
            "maxLength":20
         },
         "val1":{
            "type":"integer",
            "maximum":4294967295,
            "minimum":0
         },
         "val2":{
            "type":"integer",
            "maximum":4294967295,
            "minimum":0
         },
         "web_pic1":{
            "type":"string",
            "maxLength":20
         },
         "web_pic2":{
            "type":"string",
            "maxLength":20
         },
         "a_code":{
            "type":"string",
            "maxLength":10
         }
      },
      "required":[
         "action",
         "file_name",
         "store_id",
         "store_name",
         "loc_street",
         "loc_city",
         "loc_zip",
         "loc_country",
         "loc_rep",
         "val1",
         "val2",
         "web_pic1",
         "web_pic2",
         "a_code"
      ]
   }
```

```
         },
         "required":[
            "fstio_map"
         ]
      }
   },
   "required":[
      "FFSTIOOperationResponse"
   ]
}
```

2.2.1.3 Ecplise and IBM MobileFirst Studio

Eclipse Luna (4.4.2) and IBM MobileFirst Studio V7.0 are used for the development and deployment of the HTTP adapter.

The purpose of this adapter is to demonstrate that CICS Transaction Server for VSE/ESA and CICS Transaction Server for z/VSE V2.1 can serve HTTP(S) with JSON payloads via CICS TG V9.1 to connect to an existing CICS COMMAREA COBOL or C Language application in z/VSE.
The adapter builds and sends a HTTP(S) post request and JSON payload to the CTG daemon. The CTG daemon parses JSON payload and automatically converts the JSON data into COMMAREA data. The CTG uses ECI and passes via TCP/IP the COMMAREA data to the CICS TS running in z/VSE. A CICS TS application processes the request and generates COMMAREA data in response back to CTG. Again CTG automatically converts the COMMAREA data to JSON payload passed as HTTP response back to the adapter.

The installation and configuration of IBM MobileFirst Studio and the IBM MobileFirst Platform Foundation is described in Installing MobileFirst Studio.

After the installation, create a new project with template ***Hybrid Application*** as described in Creating MobileFirst projects. Next create a new adapter of type HTTP Adapter documented here Creating a JavaScriptMobileFirst adapter. If done expand and explore your adapter.

2.2.1.4 HTTP Adapter – XML

The member <adapter>.xml as generated needs to be changed.

The <connectivity> element must be edited with the connection data necessary to reach the CICS TG daemon. The element ‘domain’ is set to the host name of the CTG HTTP protocol handler and element ‘port’ for it’s listener. The latter value corresponds to subsection HTTP as described in 2.1.3 The CICS® Transaction Gateway configuration file - ctg.ini These lines are highlighted yellow below.

The adapter implementation procedure needs to be declared in here.

- From the original content purge the red crossed-out lines.
- Add the line in green for the declaration of procedure “postFFSToreReq”.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!--
    Licensed Materials - Property of IBM
    5725-I43 (C) Copyright IBM Corp. 2011, 2013. All Rights Reserved.
    US Government Users Restricted Rights - Use, duplication or
    disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-->
<wl:adapter name="HTTP_JSON_CTG"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wl="http://www.ibm.com/mfp/integration"
  xmlns:http="http://www.ibm.com/mfp/integration/http">

  <displayName>HTTP_JSON_CTG</displayName>
  <description>HTTP_JSON_CTG</description>
  <connectivity>
    <connectionPolicy xsi:type="http:HTTPConnectionPolicyType"
cookiePolicy="IGNORE_COOKIES">
      <protocol>http</protocol>
      <domain>vseh11.boeblingen.de.ibm.com</domain>
      <port>2080</port>
      <connectionTimeoutInMilliseconds>30000</connectionTimeoutInMilliseconds>
      <socketTimeoutInMilliseconds>30000</socketTimeoutInMilliseconds>
      <maxConcurrentConnectionsPerNode>50</maxConcurrentConnectionsPerNode>
      <!-- Following properties used by adapter's key manager for choosing specific
certificate from key store
      <sslCertificateAlias></sslCertificateAlias>
      <sslCertificatePassword></sslCertificatePassword>
      -->
    </connectionPolicy>
  </connectivity>

  <procedure name="postFFSToreReq" connectAs="server"/>
  <procedure name="getStories"/>
  <procedure name="getStoriesFiltered"/>
</wl:adapter>
```

2.2.1.5 HTTP Adapter Implementation <adapter>-impl.js - Sample HTTP POST Request and Sample JSON Payload

Replace the implementation of your <Adapter>-impl.js by the sample Javascript function below.

The Javascript function **postFFSToreReq** contains the sample JSON payload and the invocation of the HTTP service **WL.Server.invokeHttp.**

In a nutshell the FFStore action **GETKEYGE** retrieves the company record, which matches or is higher than the key provided. The key-value given for ‘store_id’ is of type string, but the value passed should be a valid integer value. The file name used for ‘FFSTORE’ is the VSAM file name from z/VSE.

The value ‘json/FFST’ given for **path** of **http_input** corresponds to the **URI** value defined in the WEBSERVICE section named FFST in the ctg.ini as shown in Chapter *2.1.3 The CICS® Transaction Gateway configuration file - ctg.ini .*

The other values set are self explanatory as required for the HTTP protocol data passed to the CICS TG daemon. The variable **http_input** keeps those definitions as JSON block parameters.

```
function postFFSToreReq() {
	/*
	 * Action codes used by FFSTIO
	 */
	const ACTION_GETKEYGE = 1; // get info of store that is identified by
a key greater or equal the given one
	const ACTION_GETNEXT  = 2; // ... next
	const ACTION_GETPREV  = 3; // ... previous
	const ACTION_UPDATE   = 4; // update info ...
	const ACTION_DELETE   = 5; // delete ...
	const ACTION_INSERT   = 6; // insert ...

	var request = {
			"FFSTIOOperation" : {
				"fstio_map" : {
					"action" : ACTION_GETKEYGE,
					"file_name" : "FFSTORE",
					"store_id" : "000029"
				}
			}
		}

	var http_input = {
		method : 'post',
		returnedContentType : 'json',
		path : 'json/FFST',
		headers : {
			'Content-Type' : 'application/json'
		},
		body : {
			contentType : 'json',
			content : JSON.stringify(request)
		}
	};

	var response = WL.Server.invokeHttp(http_input);
	return response;
}
```

2.2.1.6 RunAs – Testing the HTTP Adapter

1. Make sure the MobileFirst Development server from your MobileFirst Studio is started.
2. From the Project Explorer view expand your MobileFirst adapter
   1. Select the adapter, <right-click>, select RUN-AS, choose Call MobileFirst Adapter
   2. Window Call MobileFirst Procedure pops up with prefilled:
      Procedure name: PostFFSToreReq()
      Rest Call Type: Get
   3. Click: RUN
3. In return a browser window shows the JSON Response as received from the CTG like the one below:

JSON Response received from FFSTORE


```
{
   "FFSTIOOperationResponse": {
      "fstio_map": {
         "a_code": "password",
         "action": 1,
         "file_name": "FFSTORE",
         "loc_city": "a city name",
         "loc_country": "a county name",
         "loc_rep": "a person name",
         "loc_street": "a street name",
         "loc_zip": "a zip num",
         "store_id": "000029",
         "store_name": "a store name",
         "val1": 3000,
         "val2": 1500,
         "web_pic1": "Map.gif",
         "web_pic2": "Store20.gif"
      }
   },
   "isSuccessful": true,
   "responseHeaders": {
      "Content-Length": "328",
      "Content-Type": "application\/json",
      "Date": "Tue, 28 Jul 2015 11:38:37 GMT",
      "X-Powered-By": "Servlet\/3.0"
   },
   "responseTime": 344,
   "statusCode": 200,
   "statusReason": "OK",
   "totalTime": 349
}
```


3 Web Links and Publication References

The references listed in this section are considered by the author particularly suitable for a more broad and detailed discussion of the topic covered in this paper.

- **CICS Transaction Gateway for Multiplatforms 9.1 (IBM Knowledge Center)**
- **IBM MobileFirst Platform Foundation V7.1 documentation (IBM Knowledge Center)**
- **IBM z/VSE (Home)**
- **CICS Transaction Server (z/VSE Home)**
- **CICS Transaction Server for VSE/ESA Enhancements Guide**
- **Getting started with mobile development for z/VSE**

IBM Redbooks

The following IBM Redbook publications provide additional information about the topic in general. Note that some publications referenced in this list might be available in softcopy only.

- IBM System z in a Mobile World SG24-8215-00
- WebSphere Application Server V8.5 Concepts, Planning, and Design Guide SG24-8022
- z/VSE Basics SG24-7436
- e-business Connectivity for VSE/ESA SG24-5950
- e-business Solutions for VSE/ESA SG24-5662
- CICS Transaction Server for VSE SG24-5997
  - CICS Web support
- Servlet and JSP Programming SG24-5755

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website: http://www.redbooks.ibm.com/

Contact:
You can reach z/VSE under the following email address: zVSE@de.ibm.com

4 Appendix - Setup CICS Web Support in VSE

The following steps have to be done in order to enable CICS Web Support (CWS).

- Set up these parameters in DFHSITSP for the CICS TS your region. For example, DBDCCICS to enable CWS:
  - ISC=YES Intersystem communication enabled
  - TCPIP=YES TCP/IP protocol enabled
- Configure and enable codepage conversion in CICS. The IBM provided skeleton DFHCNV (ICCF library 59) can be adapted and submitted.
- Configure and enable the CICS Web Error program. The IBM provided skeleton DFHWBEP (ICCF library 59) has been adapted and submitted.

(FFST)

- The BMS map for application FFST was compiled with option SYSPARM='TEMPLATE'. This produces a HTML template that was adapted and stored in PRD2.DFHDOC.
  Please see the following job how to generate the template:

```
* $$ JOB JNM=FFSTMAP,DISP=D,CLASS=A,NTFY=YES
* $$ LST DISP=D,CLASS=Q,PRI=3
// JOB FFSTMAP COMPILE PROGRAM FFSTMAP
.....
#/ JOB FFSTMAP CATALOG MAP FFSTMAP
.....
#/ JOB FFSTMAP CATALOG HTML  FFSTMAP
// EXEC LIBR   ACCESS SUBLIB=PRD2.DFHDOC
* $$ END
// ON $CANCEL OR $ABEND GOTO ENDJ3
// OPTION NOLIST,ALIGN,DECK,SYSPARM='TEMPLATE'
// EXEC ASMA90,SIZE=(ASMA90,64K),PARM='EXIT(LIBEXIT(EDECKXIT)),
               SIZE(MAXC-200K,ABOVE)'
   PRINT NOGEN
* $$ SLI MEM=FFSTMAP.A,S=PRIMARY.WKS
/*
/. ENDJ3
// EXEC IESINSRT
/*
#&
$ $$ EOJ
* $$ END
/. ENDM
/& * $$ EOJ
```

- The CICS startup job has to be adapted to include the library PRD2.DFHDOC in the LIBDEF. This is required to allow CICS to find the HTM templates.

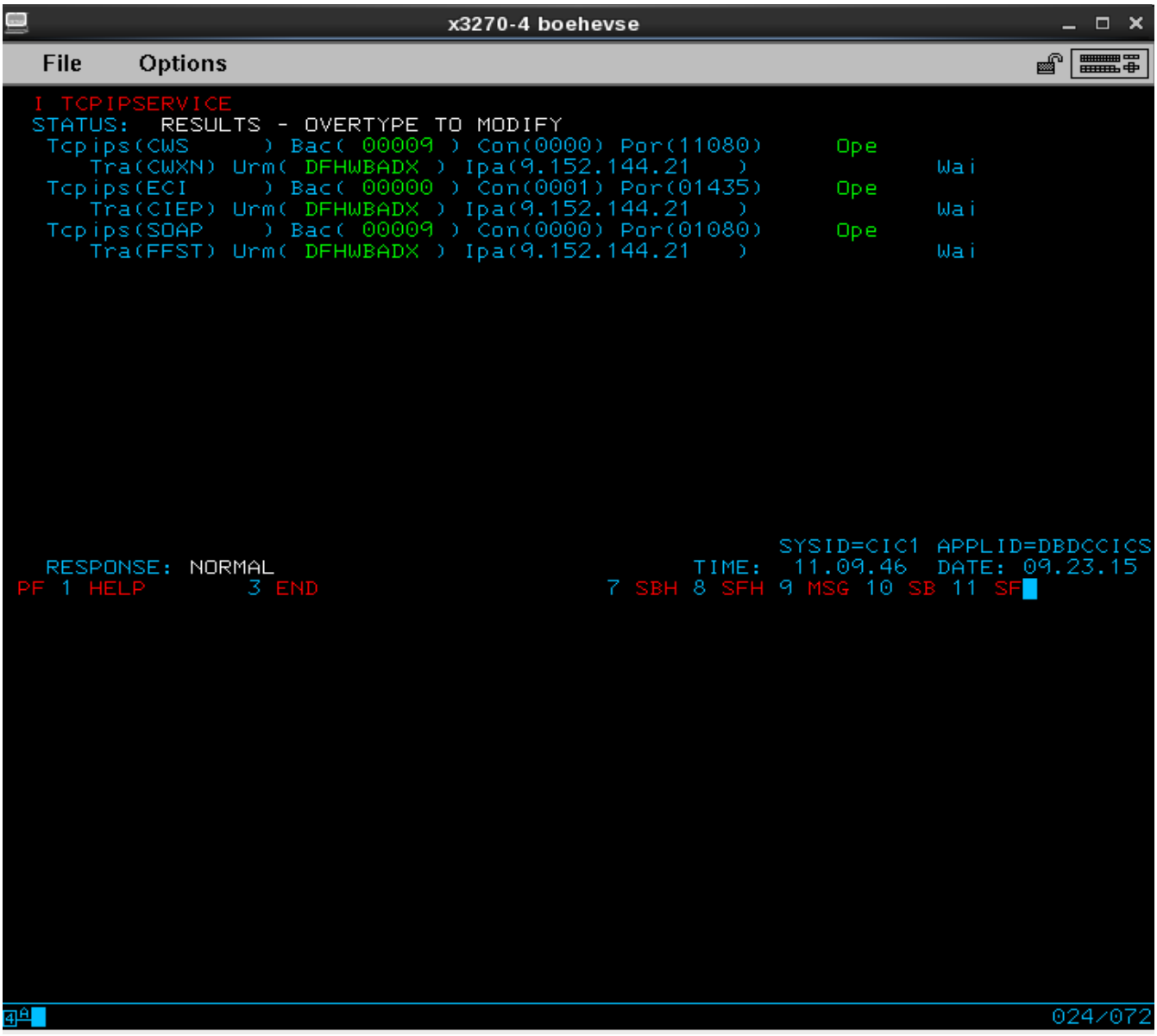
5 Appendix - TCPIP Service definition for CICS Transaction Gateway (ECI)

To allow incoming CICS requests from remote sites using CICS Transaction Gateway through External Call Interface (ECI), the CICS Web Support interface must be set up. An additional TPC/IP service must be defined with the Port for ECI requests (1435) and the associated initial transaction name (CIEP).

The TCP/IP service definition parameters are as follows:

```
CEDA  DEFine TCpipservice( ECI      )
  TCpipservice   : ECI
  Group          : VSESPG
  Description  ==> SERVICE FOR ECI
  Urm          ==>
  Portnumber   ==> 01435              1-65535
  Certificate  ==>
  STatus       ==> Open               Open | Closed
  SSl          ==> No                 Yes | No | Clientauth
  Attachsec    ==> local              Local | Verify
  TRansaction  ==> CIEP
  Backlog      ==> 00001              0-32767
  TSqprefix    ==>
  Ipaddress    ==>
  SOcketclose  ==> No                 No | 0-240000
```

When a TCP/IP product is installed on z/VSE, you can check the configuration and status of the CICS TCPIP Services running in z/VSE. The screen shot is for transaction **CEMT Inquire TCPIPService** which shows also the ECI service. This one is used in in CICS TG - Basic Configuration


x3270-4 boehevse
File
Options
I TCPIPSERVICE
STATUS: RESULTS - OVERTYPE TO MODIFY
Tcpips(CWS) Bac(00009) Con(0000) Por(11080) Ope
Tra(CWXN) Urm(DFHWBADX) Ipa(9.152.144.21) Wai
Tcpips(ECI) Bac(00000) Con(0001) Por(01435) Ope
Tra(CIEP) Urm(DFHWBADX) Ipa(9.152.144.21) Wai
Tcpips(SOAP) Bac(00009) Con(0000) Por(01080) Ope
Tra(FFST) Urm(DFHWBADX) Ipa(9.152.144.21) Wai
SYSID=CIC1 APPLID=DBDCCICS
RESPONSE: NORMAL TIME: 11.09.46 DATE: 09.23.15
PF 1 HELP 3 END 7 SBH 8 SFH 9 MSG 10 SB 11 SF
024/072