IBM® z/TPF Data Event Starter Kit User's Guide

Updated March 6, 2015

Table of Contents

Disclaimer

This package is provided on an "as is" basis. There are no warranties, express or implied, including the implied warranties of merchantability and fitness for a particular purpose. IBM has no obligation to provide service, defect correction, or any maintenance for this package. IBM has no obligation to supply any updates or enhancements for the package to you even if such are or later become available.

Data event starter kit overview

The data event starter kit shows how the z/TPF system automatically creates data events when an application updates a z/TPFDF database and how an event consumer on a remote system can receive and process those data events.

There are two major components to this starter kit:

- The z/TPF data event (DEVT) driver, which runs on a z/TPF system.
- A sample java application (DFDLtoXMLSample), which is used with the IBM Integration Bus and runs on a Windows or Linux system (also referred to as the *remote system* in this document).

For this starter kit, the DEVT driver represents the application and updates the sample Credit database using DEVT driver commands that are entered on the z/TPF command line. The Credit database is composed of three z/TPFDF files:

- DR11ED: Top-level index that references DR12ED and DR14ED
- DR12ED: Authorized users and credit card transactions for a given credit card
- DR14ED: Security information for a given credit card

The DR12ED z/TPFDF file is the only z/TPFDF file enabled for data events. When a subfile in the DR12ED file is updated, the z/TPF system automatically captures the changed z/TPFDF logical records (LRECs). The captured LRECs are packaged as an event message data event named Credit_Trxns, event user context is added by an application enrichment program, and the resulting event message data event is sent to the event consumer over a TCP/IP connection.

No data events are created when DR11ED and DR14ED z/TPFDF files are updated because those files are not enabled for data events. For more information on the Credit database layout, see the z/TPF Data Event (DEVT) Driver User's Guide (DEVTDriver_UsrGde.pdf).

The DFDLtoXMLSample represents the event consumer, reads data events from the TCP/IP connection, and creates an XML document for each data event that is received from the z/TPF system. To create the XML document, the DFDLtoXMLSample application uses the IBM DFDL parser in the IBM Integration Bus along with DFDL schema files to parse the binary data event that is received from z/TPF. By using the IBM DFDL parser and schema files, the DFDLtoXMLSample application can identify and format all of the structures and fields in the binary data event and create an XML document for the data event.

The DFDLtoXMLSample Java application saves both the binary data event and the XML document as files in the remote system's file system. Both files contain the complete Credit_Trxns data event, including the event header, event user context, and event data.

Requirements and restrictions

- The z/TPF system must have the following release levels
  - z/TPF at PUT 11 or later and with APAR PJ42834 installed
  - z/TPFDF at PUT 11 or later and with APAR PI32332 installed

- maketpf build tools for z/TPF

- The z/TPF data event (DEVT) driver must be installed and configured on the z/TPF system. For more information about how to install and configure the DEVT Driver, see the readme file (DEVTReadme.txt or DEVTReadme.pdf).

- IBM® Integration Bus 9.0 or later (http://www.ibm.com/software/products/en/ibm-integration-bus) must be installed and configured on the remote system. For information about how to install and configure Integration Bus, see the product documentation

Installing the starter kit files

1. Transfer the starter kit compressed file (DEVTStarterKit.zip) to your system. This can be placed in any directory as a holding location, for example, `C:\temp\ztpfzip`
2. Create a directory that will hold the uncompressed files, for example, `C:\ztpf\DEVTStarterKit`
3. Extract the files from the compressed file by using a compression utility. The data event starter kit files will be extracted in the following directory structure:
    - `DFDLtoXMLSample_Project.zip`
    - `bin/DFDLtoXMLSample`
    - `bin/DFDLtoXMLSample.jar`

Setting up the DFDLtoXMLSample Java application

The DFDLtoXMLSample Java application can be set up as an IBM Integration Bus application or as an independent DFDL application. Based on your environment, choose one of the methods below to set up the DFDLtoXMLSample Java application.

Instructions for running the application are in later sections after all of the setup is complete.

Setting up as an IBM Integration Bus application

The easiest way to set up the DFDLtoXMLSample Java application as an IBM Integration Bus application is to import the DFDLtoXMLSample project into the IBM Integration Toolkit. By importing the application into the IBM Integration Toolkit, the application can be easily run from the toolkit without needing to set up other systems or environments.

1. In the IBM Integration Toolkit, import the DFDLtoXMLSample_Project.zip file using the "Project Interchange" source type.
    a. From the toolkit menu, select **File->Import**.
    b. Select **Other->Project Interchange** for the import type.
    c. Follow the prompts to import the DFDLtoXMLSample_Project.zip file that was extracted in step 3 of the "Installing the starter kit files" section of this document".
    d. Select the DFDLtoXMLSample project when prompted.

For more information about importing projects, see the IBM Integration Toolkit documentation.

Setting up as an independent DFDL application

The DFDLtoXMLSample Java application can be set up to run as an independent DFDL application.

1. Refer to the IBM Integration Bus documentation about "Developing independent DFDL applications". This documentation describes how to configure a remote system that does not have IBM Integration Bus broker component installed.

2. In step 3 of the "Installing the starter kit files" section of this document, some of the starter kit files were extracted to a `bin/` directory. Transfer the data event starter kit files provided in the `bin/` directory to a directory on the remote system where you want to run the DFDLtoXMLSample Java application, for example, `/ztpf/DEVTStarterKit/bin`

3. Edit the DFDLtoXMLSample script as follows:
    a. Update the `SampleJARDir` variable to refer to the directory where the DFDLtoXMLSample.jar file was saved in step 2.
    b. Part of step 1 above includes setting up IBM DFDL JAR files on the remote system. Update the `DFDLLibDir` variable in the script to refer to the directory where you installed the IBM DFDL JAR files as part of step 1.

4. Verify that the file permissions for the DFDLtoXMLSample script allow both read and execute permissions. For example, issue the following command to set read and execute permissions for all users:
    a. `chmod 755 DFDLtoXMLSample`

Setting up directories and DFDL schema files

The DFDLtoXMLSample Java application uses DFDL schema files to parse the binary data event and create an XML document. The DFDL schema files from the z/TPF product and the z/TPF data event driver must be copied to a directory where they can be read by the DFDLtoXMLSample Java application.

1. On the remote system where you will run the DFDLtoXMLSample Java application, create a directory to hold the DFDL schemas that will be used by the application, for example, `C:\ztpf\DEVTStarterKit\schemas`

2. Transfer the following DFDL schemas that are provided with the z/TPF product to the directory that you created in step 1.
   - `base/tpf-fdes/cdev.lib.dfdl.xsd`
   - `base/tpf-fdes/ibev.lib.dfdl.xsd`
   - `base/tpf-fdes/tpfbase.lib.dfdl.xsd`
   - `base/tpf-fdes/ubev_ecbctx.lib.dfdl.xsd`

3. Transfer the following DFDL schemas that are provided with the z/TPF data event (DEVT) driver to the directory that you created in step 1. These files were installed on your Linux system when you installed the DEVT driver.
   - `devt/schemas/Credit_Trxns.de.dfdl.xsd`
   - `devt/schemas/Credit_Trxns_Usrctx.user.dfdl.xsd`
   - `devt/schemas/DR12ED.tpfdf.dfdl.xsd`

4. On the remote system where you will run the DFDLtoXMLSample Java application, create a directory where the application can save the binary and XML formatted data event files, for example, `C:\ztpf\DEVTStarterKit\output`

Running the DFDLtoXMLSample Java application

The following instructions describe how to start the DFDLtoXMLSample Java application. See Example 1 on page 11 for an example of the output received when starting the application.

1. Start the DFDLtoXMLSample Java application in one of the following ways:
    a. If you are running the DFDLtoXMLSample Java application from the IBM Integration Toolkit, complete the following steps:
        i. From either the **Java** or **Integration Development** perspective in the IBM Integration Toolkit, right click on the DFDLtoXMLSample project and choose **Run as->Java Application**.

        ii. Select **DFDLtoXMLMain** if prompted for a Java application or entry point.

        iii. When the DFDLtoXMLSample Java application starts, prompts will appear in the IBM Integration Toolkit console requesting input.

    b. If you are running the DFDLtoXMLSample Java application as a stand-alone DFDL application, do the following:
        i. Change to the directory where you saved the DFDLtoXMLSample bin files; for example, enter `cd /ztpf/DEVTStarterKit/bin`

        ii. Start the application by executing the DFDLtoXMLSample script; for example, enter `DFDLtoXMLSample`

        iii. When the DFDLtoXMLSample Java application starts, prompts will appear in the console where you started the application.

2. When the DFDLtoXMLSample Java application is started, it begins by requesting input.
    a. You will be asked for the directory where the DFDL schema files are located. This is the directory that was created in step 1 of the “Setting up directories and DFDL schema files” section of this document.

    b. You will be asked for the output directory where you want the application to save files. This is the directory that was created in step 4 of the “Setting up directories and DFDL schema files” section of this document.

    c. You will be asked for the socket port number that the application should read data events from. This must be the same port number set in the `devt/eventspecs/IPSendBINAdapter.evda.xml` business event dispatch adapter for the z/TPF data event (DEVT) driver.

3. The DFDLtoXMLSample Java application reads the DFDL schema files and creates a grammar based on the schemas. The grammar is used by the IBM DFDL parser to parse each data event as it is received. After creating the grammar, the application opens a socket by using the port number that was provided and reads from the socket, waiting for data events to arrive from your z/TPF system.

Example 1: Starting the DFDLtoXMLSample Java application

```
Enter the directory where the DFDL schema files are located: C:\ztpf\DEVTStarterKit\schemas
Enter the output directory to save files in: C:\ztpf\DEVTStarterKit\output
Enter the socket port number to read from: 9876

Creating DFDL grammar for schema C:/ztpf/DEVTStarterKit/schemas/Credit_Trxns.de.dfdl.xsd
Creating DFDL parser for the DFDL grammar.
DFDL grammar and parser setup completed successfully.

Socket established on port 9876.
Waiting on socket connection from client...
```

Creating data events

When the z/TPF data event (DEVT) driver is used to update the z/TPFDF credit database on z/TPF, data events are created automatically and sent to the remote system. For more information on the syntax and use of the DEVT Driver, see the z/TPF Data Event (DEVT) User's Guide. To get started quickly, the following DEVT driver commands can be used to generate data events.

See Example 2 on page 14 for an example of the output received when DFDLtoXMLSample receives and processes a data event.

The commands listed below assume the CREDIT database has been initialized and built with predefined data.

1. Display the users for credit card 1895472010006809. This command does not generate any data events because it does not update the database.

   User:
   ```
   ztest devt credit display user brand-0 cardnum-1895472010006809
   ```

   System:
   ```
   DEVT0523I CREDIT USER DISPLAY:
     MATCHING USER(S) FOR BRAND-0 CARDNUM-1895472010006809
       NAME-ROSIEARMSTRONG       SSN-710230865
       BILLING ADDRESS-11 CORTES RD
       SECURITY INFO: YES        STATUS: NONE
      --- END OF USER INFO
     END DISPLAY USERS FOR BRAND-0 CARDNUM-1895472010006809
   END OF DISPLAY
   ```

2. Add a new user to credit card 18954720100068099. This command generates a data event that contains one LREC with LREC ID X'D3'. Optionally, the command in step 1 can be used to show both users.

   User:
   ```
   ztest devt credit add user brand-0 cardnum-1895472010006809 name-
   BILLARMSTRONG address-'11 CORTES RD' ssn-123456789
   ```

   System:
   ```
   DEVT0572I 11.31.28 USER ADDED TO CREDIT DATABASE.
   ```

3. Display the purchases for credit card 1895472010006809. This command does not generate any data events because it does not update the database.

   User:
   ```
   ztest devt credit display charge brand-0 cardnum-1895472010006809
   ```

   System:
   ```
   DEVT0521I CREDIT CHARGE DISPLAY:
      CHG FOR BR-0 CARDN-1895472010006809 NAME-ROSIEARMSTRONG       SSN-710230865
               VENDOR-012345678 DATE-2007JUL31 TIME-12:05 AMT-$9.13
      CHG FOR BR-0 CARDN-1895472010006809 NAME-ROSIEARMSTRONG       SSN-710230865
               VENDOR-005566777 DATE-2007FEB12 TIME-20:20 AMT-$168.50
      CHG FOR BR-0 CARDN-1895472010006809 NAME-ROSIEARMSTRONG       SSN-710230865
               VENDOR-000555555 DATE-2007MAY24 TIME-21:21 AMT-$17.45
      --- END OF CHARGES FOR CREDIT CARD
   END OF DISPLAY
   ```

4. Add a new purchase for the user that was added in step 2. This generates a data event that contains one LREC with LREC ID X'D7'. Optionally, the command in step 3 can be used to show all credit card purchases for this card.

   User:
   ```
   ztest devt credit add charge brand-0 cardnum-1895472010006809 ssn-123456789
   vendor-100012345 date-2015FEB05 time-09:27 amt-45.63
   ```

   System:
   ```
   DEVT0578I 11.35.01 CHARGE ADDED.
   ```

5. Delete all purchases for credit card 1895472010006809. This generates a data event that contains four LRECs, all with LREC ID X'D7'.

   User:
   ```
   ztest devt credit delete charge brand-0 cardnum-1895472010006809
   ```

   System:
   ```
   DEVT0511I 11.54.16 CREDIT CHARGE DELETE COMPLETE.  FOLLOWING DELETED:
      CHG FOR BR-0 CARDN-1895472010006809 NAME-ROSIEARMSTRONG       SSN-710230865
               VENDOR-12345678 DATE-2007JUL31 TIME-12:05 AMT-$9.13
      CHG FOR BR-0 CARDN-1895472010006809 NAME-ROSIEARMSTRONG       SSN-710230865
               VENDOR-5566777 DATE-2007FEB12 TIME-20:20 AMT-$168.50
      CHG FOR BR-0 CARDN-1895472010006809 NAME-ROSIEARMSTRONG       SSN-710230865
               VENDOR-555555 DATE-2007MAY24 TIME-21:21 AMT-$17.45
      CHG FOR BR-0 CARDN-1895472010006809 NAME-BILLARMSTRONG        SSN-123456789
               VENDOR-100012345 DATE-2015FEB05 TIME-09:27 AMT-$45.63
      --- END OF CHARGES DELETED FOR CREDIT CARD
   END OF DISPLAY
   ```

Example 2: Output from the DFDLtoXMLSample Java application when receiving and processing one data event

```
Enter the directory where the DFDL schema files are located: C:\ztpf\DEVTStarterKit\schemas
Enter the output directory to save files in: C:\ztpf\DEVTStarterKit\output
Enter the socket port number to read from: 9876

Creating DFDL grammar for schema C:/ztpf/DEVTStarterKit/schemas/Credit_Trxns.de.dfdl.xsd
Creating DFDL parser for the DFDL grammar.
DFDL grammar and parser setup completed successfully.

Socket established on port 9876.
Waiting on socket connection from client...connection established.

Waiting to receive message...
Size of incoming message is 435 bytes.
Message read successfully.
Binary format event message written to C:/ztpf/DEVTStarterKit/output/Credit_Trxns_1.bin
Parsing binary event message and creating an XML document.
XML format event message written to C:/ztpf/DEVTStarterKit/output/Credit_Trxns_1.xml

Waiting to receive message...
```