Tutorial 9

Aggregation of CICS Transactions with the Service Flow Feature

Erstellen eines Web Services unter Verwendung des im WebSphere Developer für System z Version 7.0 integriertem Service Flow Features.

Vorwort

Dieses Tutorial über den Service Flow Modeler basiert auf dem CICS Catalog Manager und demonstriert, wie die im Service Flow Feature integrierten Service Flow Project Tools verwendet werden können, um aus einer CICS Terminal Applikation einen Service Flow zu generieren, welcher anschließend als Web Service im CICS läuft.

Für dieses Tutorial wird der WebSphere Developer für System z Version 7.0 (WDz) verwendet um auf dem z/OS System[1] der Universität Leipzig[2] die nötigen Ressourcen zu definieren, einen Web Service zu erstellen, den erforderlichen Code zu generieren und letztendlich den Web Service zu testen. Der für dieses Tutorial verwendete WDz wurde auf einem virtuellen Rechner der Universität Leipzig installiert.

Dieses Tutorial liefert eine detaillierte Anleitung um einen nicht trivialen Geschäftsprozess zu erstellen und gibt dabei einen Überblick über viele Schlüsselfunktionen der Service Flow Project Tools. So wird beispielsweise:

- ein Projekt erstellt,
- die Bildschirmdefinitionen der verwendeten Screens erstellt,
- ein Flow aufgezeichnet,
- die Verwendung des Flow Editors demonstriert,
- der Nutzen von Messages erklärt,
- die Handhabung der einzelnen Screens der Terminal Applikation erklärt,
- der Runtime Code generiert und deployed,
- der Geschäftsprozess als Web Service ins CICS integriert
- und anschließend wird dieser Geschäftsprozess im Web Services Explorer ausgeführt, bzw. getestet.

Dabei wird ausführlich, Schritt für Schritt, beschrieben, wie der Code auf dem Host in Leipzig generiert und deployed wird.

---

[1] mit ADCD z/OS 1.8

[2] IP-Adresse des Hosts: 139.18.4.34

Nachdem bekannt ist, wie der CICS Catalog Manager grundlegend aufgebaut ist, soll dieser nun mit Hilfe des CICS Web Service Support modernisiert werden. Zwei Gründe die für eine Modernisierung sprechen könnten z.B. einerseits die Beschränkung der Datenlänge des CTG auf 32k sein, oder andererseits der Wunsch nach einem Web Service Interface sein. Dieses Interface entkoppelt den Client vom Backend und ermöglicht so einen Zugriff auf eine Anwendung ohne Hintergrundwissen über die genaue Implementierung zu benötigen.

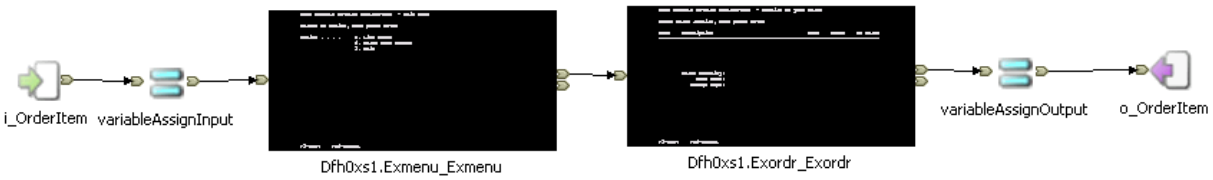


Abbildung 1

Um den in Abbildung 1 dargestellten Service Flow zu erstellen sind die folgenden Schritte nötig:

1. Einloggen auf dem Windows XP-Server mit der WDz 7 Installation
2. Ändern der Spracheinstellungen
3. WDz starten
4. Anlegen einer Host Verbindung
5. Anlegen der benötigten Ressourcen
6. Ein neues Service Flow Projekt erstellen
7. Vorbereitung: Screen Definitionen importieren
8. Flow Aufzeichnen
9. Erstellen und Bearbeiten des Generation Properties Files
10. Generieren und Deployen des Runtime Codes
11. Überprüfen und Bearbeiten des erstellten Runtime Codes
12. Installation des Webservices
13. Testen des Webservices mit dem Web Services Explorer

1. Einloggen auf dem Windows XP-Server mit der WDz 7 Installation

1. Es wird die Windows Remote Desktop-Verbindung verwendet, um Zugang zu dem Windows XP-Server zu bekommen. Dieser besitzt die IP-Adresse **139.18.8.212** (Abbildung 2).

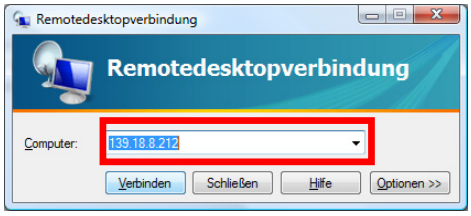


Abbildung 2

2. Anschließend mit folgenden Parametern anmelden (Abbildung 3):

   Benutzername: ibm7p01
   Kennwort: allmingo

Abbildung 3

2. Ändern der Spracheinstellungen

Da unter Verwendung der deutschen Spracheinstellungen für den WDz immer ein Fehler bei der Erstellung der WSDL auftritt, muss die Sprache vom WDz von deutsch auf englisch umgestellt werden.

Unter **Start → Systemsteuerung → Regions- und Spracheinstellungen** die Sprache auf **Englisch (USA)** stellen und anschließend Übernehmen (Abbildung 4).

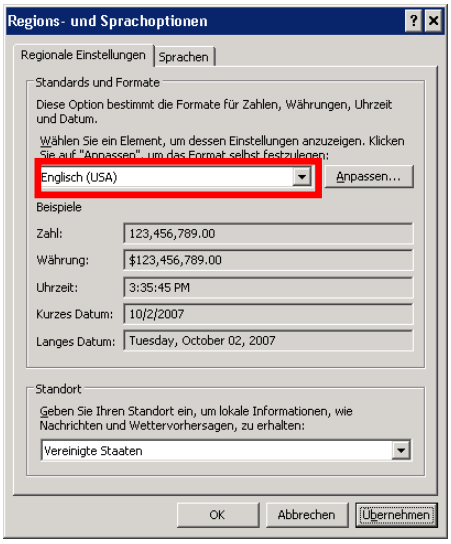


Abbildung 4

3. WDz starten

1. Nachdem die Windows Oberfläche erschienen ist, muss der Wdz gestartet werden (Abbildung 5):

   Start → Alle Programme → IBM Software Development Platform → IBM WebSphere Developer für System z → IBM WebSphere Developer für System z

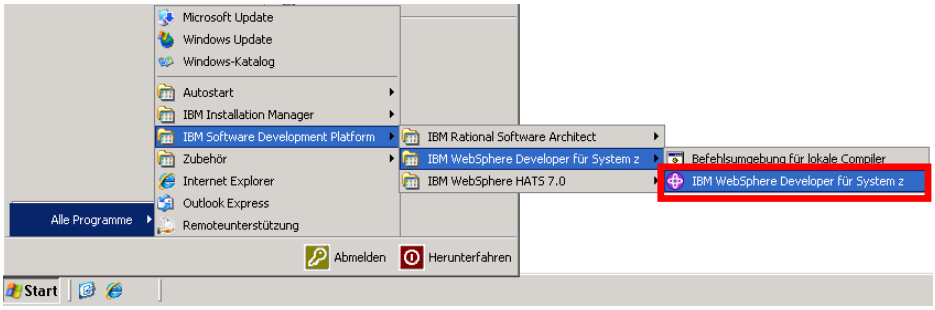


Abbildung 5

2. Anschließend wird gefragt, unter welchem Pfad der Arbeitsbereich (Workspace) angelegt werden soll. Dieser soll in unserem Beispiel unter **C:\ibm7p01\workspace** angelegt werden (Abbildung 6). Anschließend startet mit einem Klick auf **OK** die WDz Oberfläche.

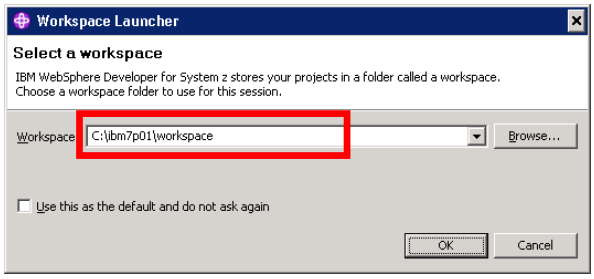


Abbildung 6

3. Beim erstmaligen Starten des WDz erscheint das Fenster „*Welcome*“. In diesem kann die Oberfläche, bzw. die zur Verfügung stehende Funktionalität der WDz Werkzeuge an verschiedest Benutzerprofile angepasst werden. Für dieses Tutorial muss die Rolle des „*z/OS Modernization Developer*“ freigeschalten werden, indem man auf das Rollen-Symbol: klickt und anschließend **z/OS Modernization Developer (advanced)** auswählt (Abbildung 7).

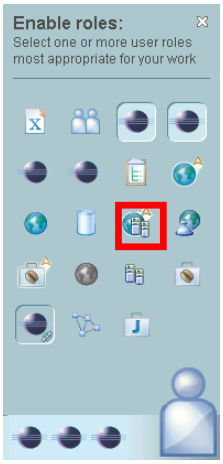


Abbildung 7

4. Das „*Welcome*“ Fenster schließen (Abbildung 8).

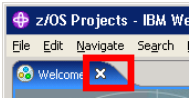


Abbildung 8

5. Es öffnet sich standardmäßig die „*z/OS-Projects*“ Perspektive.

4. Anlegen einer Host Verbindung

Hier wird beschrieben wie eine neue z/OS Verbindung zwischen dem WDz und den WDz Host Komponenten erstellt wird. Diese wird im Anschluss für das Generieren und Deployen des Codes verwendet.

1. In der Remote System Sicht im Feld New Connection mit einem Rechtsklick auf **z/OS…** → **New Connection…** auswählen (Abbildung 9).

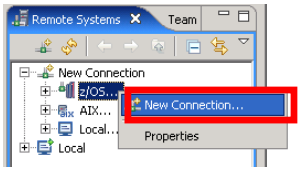


Abbildung 9

2. Es kann nun beliebig ein Profilname, Verbindungsname und eine Beschreibung gewählt werden. Allerdings sollte der Host Name mit dem Namen des z/OS Systems auf welchem die WDz Host Komponenten installiert sind übereinstimmen, bzw. auf diesen hinweisen. Als Profilname wird **wbsphere1** standardmäßig vorgeschlagen und mit **Next** übernommen. Als Host Name wird **139.18.4.34** eingetragen (Abbildung 10). Aus dieser Eingabe wird automatisch der Verbindungsname generiert. Hier: 139.18.4.34. Die restlichen Einstellungen werden auf ihren Vorgaben belassen. Nun wird mit einem Klick auf **Finish** der Wizard beendet.

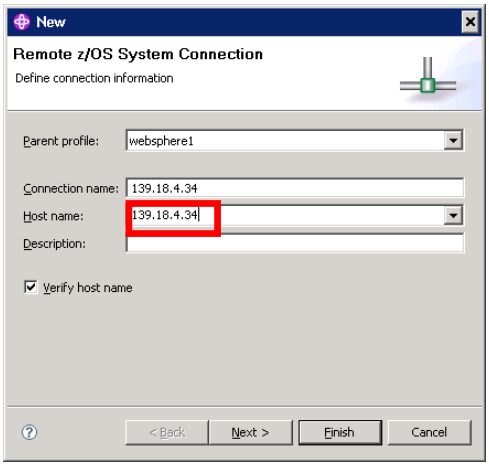


Abbildung 10

3. Nun ist es notwendig, eine Verbindung mit den WDz Host Komponenten herzustellen. Dazu ein Rechtsklick mit der Maus im Remote System Explorer auf die soeben erstellte Verbindung mit dem Namen **139.18.4.34** und **Connect** aus dem Kontextmenü auswählen (Abbildung 11).

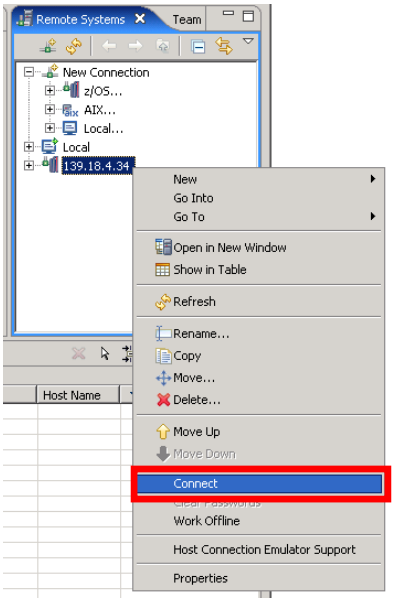


Abbildung 11

4. Für dieses Tutorial wurde für Sie ein neuer Benutzer-Account angelegt. Sie wählen hier:

   Benutzername: fstefa2
   Passwort: uni4you

   und klicken anschließend auf **OK** (Abbildung 12).

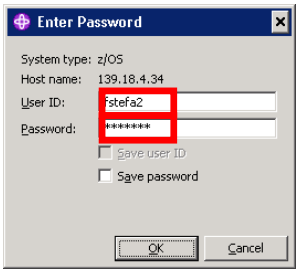


Abbildung 12

5. Sollte das Fenster mit der Warnung. dass es sich um eine Verbindung ohne SSL Verschlüsselung handelt auftauchen, so sollte die Checkbox **Do not show this message again** aktiviert werden und anschließend der Verbindungsaufbau mit **YES** bestätigt werden (Abbildung 13).

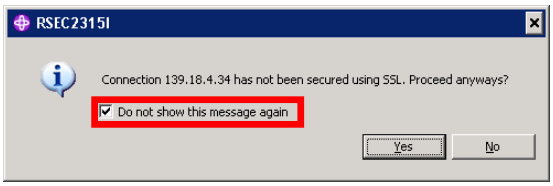


Abbildung 13

5. Anlegen der benötigten Ressourcen

Wenn nun eine Verbindung zum Host besteht, können die benötigten Ressourcen angelegt werden. Es werden die PDS für den:

- Cobol Code (FSTEFA2.USER.SRCLIB),

- die Copybooks (FSTEFA2.USER.COPYLIB)
- und die JCL's (FSTEFA2.USER.JCLLIB) angelegt,
- sowie die Verzeichnisse für die wsbind Datei (/u/FSTEFA2/wsbind)
- und wsdl Datei (/u/FSTEFA2/wsdl)im USS erzeugt.

1. Um die benötigten PDS anzulegen muss im Remote System Explorer mit einem Rechtsklick auf **MVS Files** und im darauf erscheinenden Kontextmenü **Allocate PDS…** ausgewählt werden (Abbildung 14).

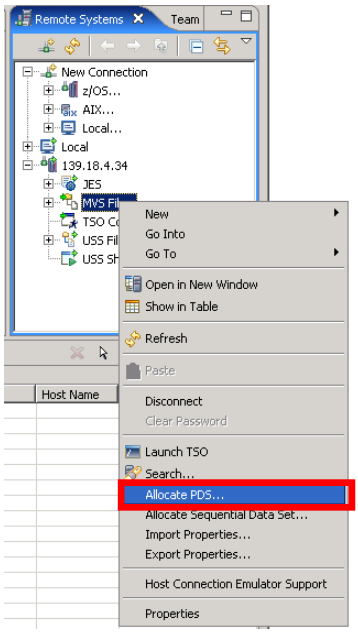


Abbildung 14

2. Als Data Set Name wird **USER.SRCLIB** eingegeben (Abbildung 15) und anschließend auf **Next** geklickt.

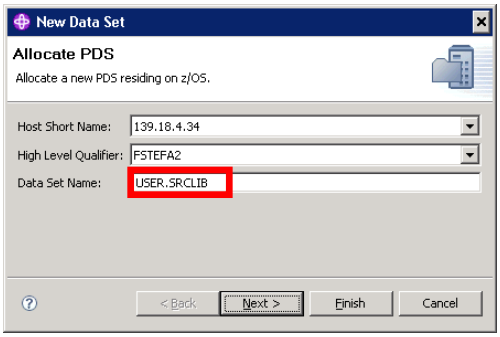


Abbildung 15

3. Im nun erscheinenden Fenster wird als Kategorie des Data Sets **SOURCE** und als Typ **COBOL** ausgewählt (Abbildung 16). Letzten Endes wird mit einem Klick auf **Finish** der PDS FSTEFA2.USER.SRCLIB angelegt.

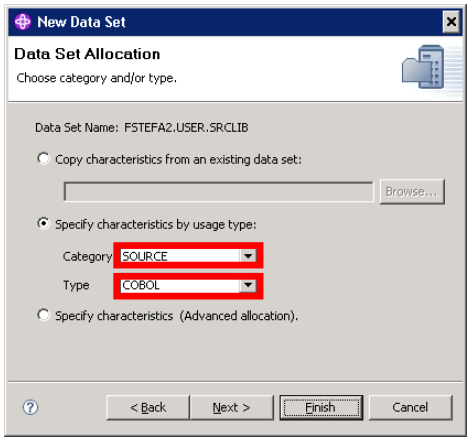


Abbildung 16

4. Um den PDS FSTEFA2.USER.COPYLIB anzulegen, muss Schritt 2 und 3 wiederholt werden, jedoch muss als Data Set Name **USER.COPYLIB** verwendet werden. Die Kategorie und der Typ sind identisch des FSTEFA2.USER.SRCLIB Data Sets.

5. Um den PDS FSTEFA2.USER.JCLLIB anzulegen, muss erneut Schritt 2 und 3 wiederholt werden, jedoch mit dem Data Set Name **USER.JCLLIB**. Anstelle des Typs COBOL wie in Abbildung 16 gezeigt, wird hier jedoch **JCL** verwendet.

Nun müssen noch die zwei USS Ordner angelegt werden, die die später erzeugte wsbind und wsdl Datei aufnehmen. Diese Verzeichnisse sollen folgende Struktur besitzen:

- /u/FSTEFA2/wsbind
- /u/FSTEFA2/wsdl

Das wsbind Verzeichnis enthält später das Web service binding file, welches später mit der Pipeline assoziiert wird. Wenn eine neue PIPELINE installiert wird oder ein entsprechender CEMT PERFORM PIPELINE SCAN durchgeführt wird, sucht CICS in dem wsbind Verzeichnis nach Dateien mit der Endung .wsbind. Für jede gefundene Datei erstellt CICS dynamisch eine WEBSERVICE und URIMAP Ressource, assoziiert diese mit der PIPELINE und installiert sie letztendlich.

6. Das wsbind Verzeichnis wird durch einen Rechtsklick auf den **My Home** Filter im USS Verzeichnis des Remote System Explorers und Auswahl von **New** → **Folder** erstellt (Abbildung 17).

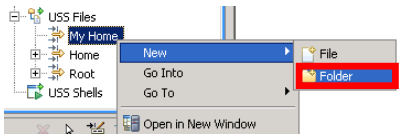


Abbildung 17

7. Als Ordnername wird **wsbind** verwendet (Abbildung 18). Mit einem Klick auf **Finish** ist dieser Ordner nun angelegt.

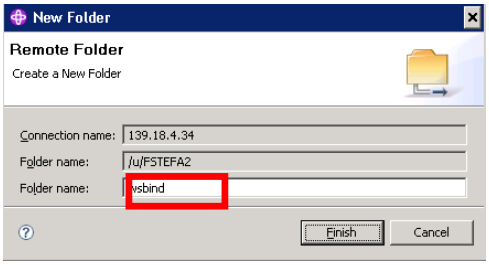


Abbildung 18

8. Nun muss Schritt 6 und 7 für den Ordner wiederholt werden, welcher später die wsdl Datei aufnimmt. Als Ordnername soll hier nun **wsdl** verwendet wird.

6. Ein neues Service Flow Projekt erstellen

In diesem Schritt wird ein neues Service Flow Projekt für den neuen Geschäftsprozess angelegt. Da bis jetzt in der „*z/OS Projects*“ Perspektive gearbeitet wurde, muss für die Erstellung des Service Flow Projektes diese nun auf die „*Enterprise Service Tools*“ Perspektive umgestellt werden.

1. Der Wechsel der Perspektive erfolgt mit:

   Window → Open Perspective → Other → Enterprise Service Tools

   und einem anschließendem Klick auf **OK**.

2. Ist die Perspektive geöffnet, so kann nun mit den folgenden Schritten ein neues Service Flow Project erstellt werden. Mit einem Rechtsklick in einen freien Bereich des „*EST Project Explorer*“ Fensters **New → Service Flow Project** auswählen (Abbildung 19). Der „*New Service Flow Project*“ Wizard öffnet sich.

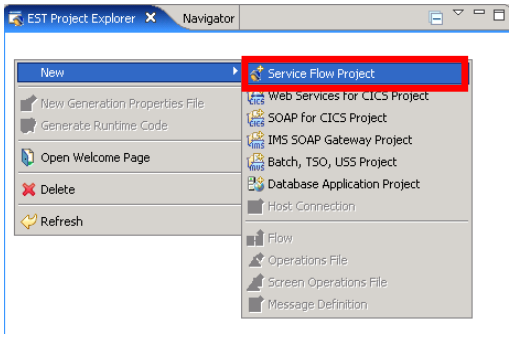


Abbildung 19

3. Im ersten Schritt des Wizards muss der Projekt Name spezifiziert werden. Wir nennen unser neues Projekt **OrderItem** (Abbildung 20) und klicken auf **Next**.

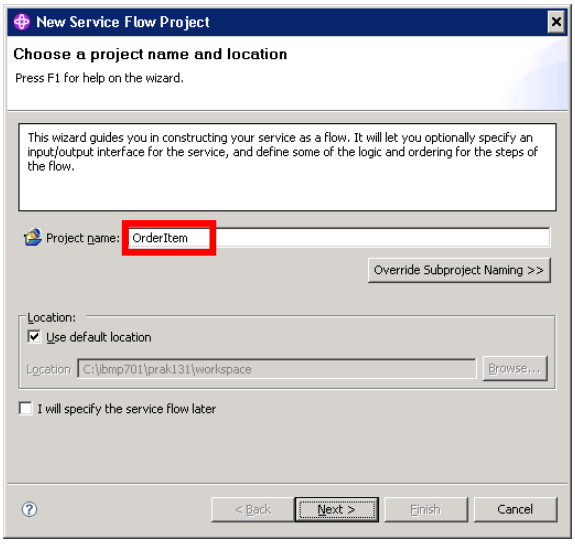


Abbildung 20

4. Im zweiten Schritt des Wizards, in welchem man das neue Projekt in bereits existierende Projekte integrieren kann, wählen wir **Define later** um später unser eigenes Interface zu generieren (Abbildung 21). Anschließend gelangt man mit einem Klick auf **Next** zum dritten Schritt des Wizards.

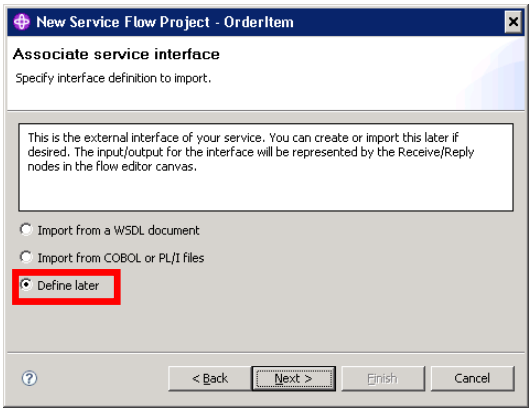


Abbildung 21

5. Im dritten Schritt muss nun noch angegeben werden, für was der Service Flow eigentlich verwendet werden soll. Hier wählen wir **Recording interactions with a terminal application** um die Interaktion mit einer Terminal Anwendung aufzuzeichnen und klicken auf **Next** (Abbildung 22). Alternativ wäre hier unter anderem möglich eine gewisse Reihenfolge an Schritten (Steps) zu definieren, die später mit einem Fluß oder andrere Logik gefüllt werden könnte.

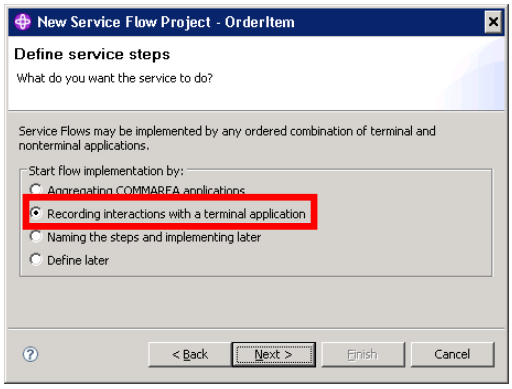


Abbildung 22

6. Im letzten Schritt des Wizards muss noch eine Host Verbindung für das Service Flow Projekt definiert werden. Als Host Name wählen wir hier die IP-Adresse **139.18.4.34** und beenden den Wizard mit einem Klick auf **Finish** (Abbildung 23).

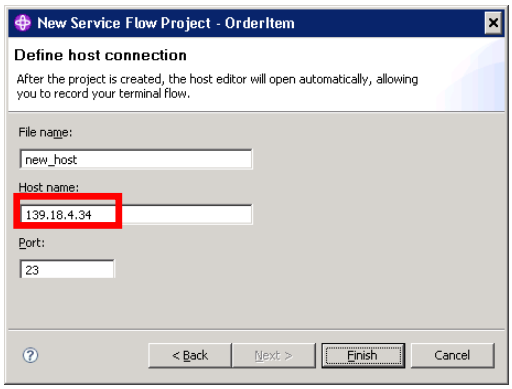


Abbildung 23

Das nun erschienene Fenster mit Enterprise Service Tool Tipps kann nach markieren der Checkbox **Do not show any tips** mit einem Klick auf **OK** geschlossen werden (Abbildung 24). In der Hauptansicht des Workspace öffnet sich nun der Hostverbindungsemulator mit einer Verbindung zur soeben eingegebenen IP-Adresse.

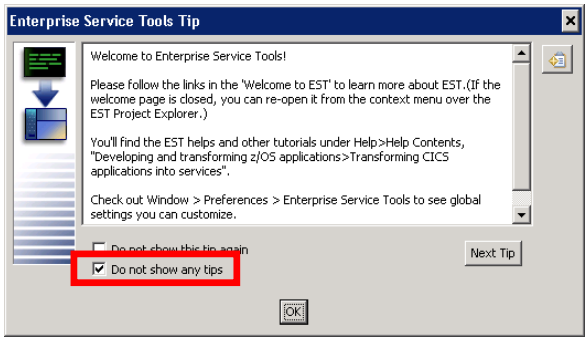


Abbildung 24

Wenn man nun das Projekt **OrderItem** im EST Project Explorer expandiert, so sieht man, dass die folgenden Ordner für das Projekt angelegt wurden:

- OrderItem.Interface
- OrderItem.Terminal
- OrderItem.Nonterminal

- OrderItem.OutboundWebService

Expandiert man nun die einzelnen Ordner, so ergibt sich folgendes Bild (Abbildung 25). Es wurde angelegt:

- eine Interface Datei: intf_OrderItem.wsdl
- drei Messages:
  - OrderItem_scratchpad.mxsd
  - i_OrderItem.mxsd
  - o_OrderItem.mxsd
- sowie eine Host Verbindung: new_host.host

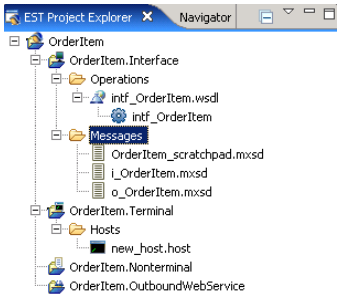


Abbildung 25

Nun kann das Aufzeichnen des Service Flows beginnen. Doch zuvor sollte noch als kleine Erleichterung, die automatische Benennung von Variablen, aktiviert werden. Dies hat den Vorteil, dass nicht bei jeder neuen Variable, also die Eingabe des Benutzers oder die Extraktion von Daten aus der Applikation, ein Name für das entsprechende Datenfeld angegeben werden muss. Dies geschieht in Zusammenarbeit mit den BMS Definitionen der Anwendungen automatisch.

Hierzu wählen wir folgendes:

Window → Preferences → Enterprise Service Tools → Service Flow Projects

Expandiert man jetzt den Eintrag **Service Flow Projects**, so sind hier alle relevanten Einstellungsmöglichkeiten für die Service Flow Tools untergebracht. Auf die für die Generierung des Runtime Codes wichtigsten, die JCL Template Einstellungen, wird später eingegangen. Wir wählen hier lediglich die Checkbox **Do not prompt for variables when generating variable mapping** aus (Abbildung 26) und klicken anschließend auf **OK**. Nach Abschluss dieses Schrittes kann mit der Aufzeichnung begonnen werden.

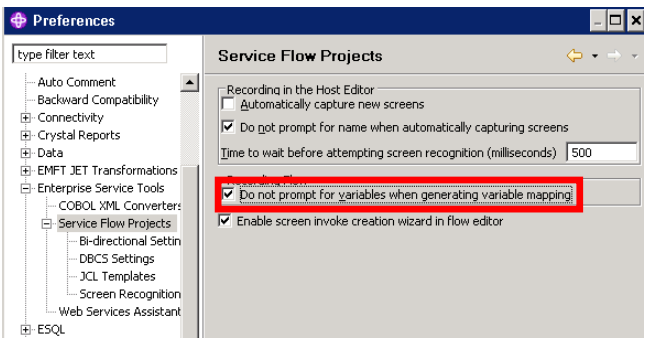


Abbildung 26

7. Vorbereitung: Screen Definitionen importieren

Damit zwischen den einzelnen Bildschirmdarstellungen (screens) unterschieden werden kann, müssen sogenannte screen Definitionen erstellt, bzw. importiert werden. Die dadurch definierten screens werden dann vom Anwendungsprogramm mit Ausgabedaten gefüllt. Die Screens bestimmen, wie und wo die Ausgabedaten im jeweiligen Fenster dargestellt werden.

1. Um die screens, die für unser CICS Catalog Manager Beispiel benötigt werden zu importieren, muss im EST Project Explorer mit der rechten Maustaste auf den Ordner **OrderItem.Terminal** geklickt werden und anschließend: **Import** → **BMS** ausgewählt werden (Abbildung 27).

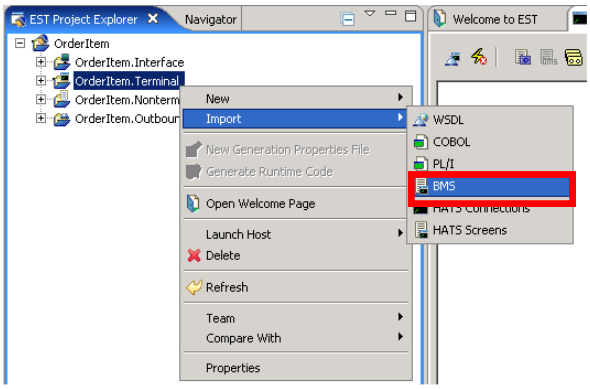


Abbildung 27

2. Im soeben geöffneten Import Wizard wählen wir Import from: **File System…**, da die benötigten BMS Dateien bereits auf dem Desktop des Rechners liegen.

   Standardmäßig sind diese unter: DFH310.CICS.SDFHSAMP auf dem Host zu finden. Dieses ist im Übrigen das Verzeichnis, in welches die CICS Catalog Manager Beispielanwendung standardmäßig kopiert wird. Von hier aus wurde die Anwendung auf dem Host installiert.

   Im Öffnen Dialog geben wir als Ordner **Desktop** an und markieren mit gedrückter Shift-Taste die beiden BMS Dateien **DFH0XS1.bms** und **DFH0XS2.bms**. Anschließend wählen wir **Öffnen** (Abbildung 28).

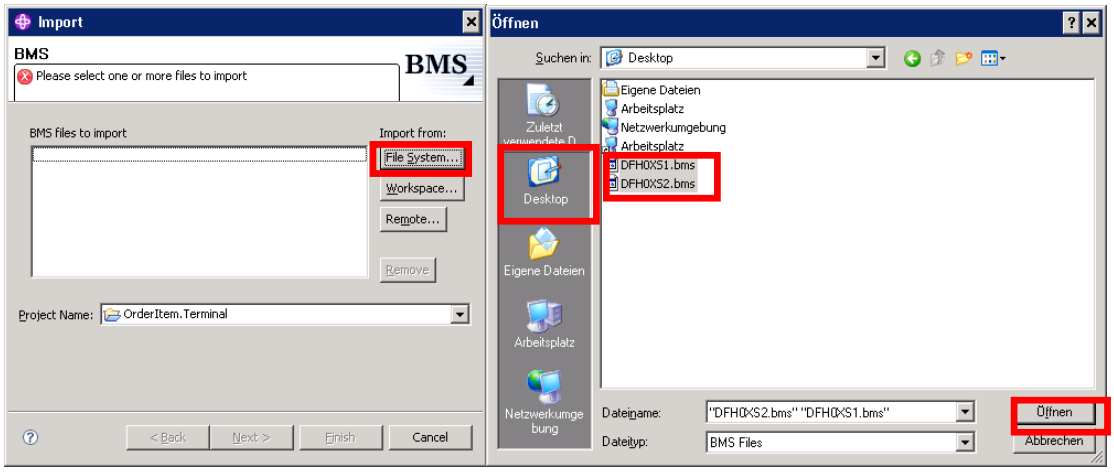


Abbildung 28

Da in diesem Beispiel nur mit einem Projekt gearbeitet wird, ist es nicht notwendig hier irgendwelche Änderungen vorzunehmen. Nachdem der Import Wizard diese beiden Dateien nun übernommen hat, kann er mit einem Klick auf **Finish** geschlossen werden (Abbildung 29).

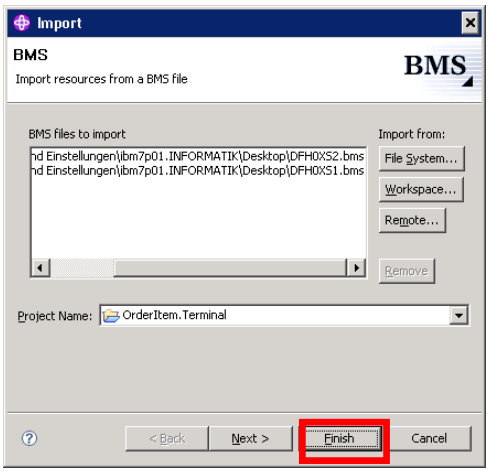


Abbildung 29

Nun wurde im „*EST Project Explorer*“ Fenster unter **OrderItem.Terminal** → **Messages** die drei screen Definitionen:

- Dfh0xs2.Exinqc.mxsd,
- Dfh0xs1.Exmenu.mxsd und
- Dfh0xs1.Exorder.mxsd erstellt.

Hier wird im XML-Format der gesamte Bildschirmaufbau des CICS Catalog Managers inkl. aller möglichen Eingabefelder beschrieben. Hier als Beispiel der Aufbau der Datei Dfh0xs1.Exmenu.mxsd, die das Hauptmenü des CICS Catalog Managers darstellt. Gelb hervorgehoben ist hier exemplarisch das Eingabefeld in welchem die „Order Item Number“, eine vierstellige Artikelnummer erwartet wird. Dieser Wert ist dann später unter der Variable ITEM-REF zu erreichen.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:Dfh0xs1_Exmenuns="http://terminal.dfh0xs1.exmenu.com/schemas"
        targetNamespace="http://terminal.dfh0xs1.exmenu.com/schemas">
- <xsd:group name="Dfh0xs1.Exmenu">
```

```
- <xsd:annotation>
- <xsd:appinfo source="WMQI_APPINFO">
  <ScreenDesc codePage="037" columns="80" messageSetDefaultRep="IBMTerminal" position="339" rows="24" />
+ <BMSDesc map="EXMENU" mapset="DFH0XS1" messageSetDefaultRep="BMSTerminal" tioaprefix="true">
      </xsd:appinfo>
      </xsd:annotation>
- <xsd:sequence>
- <xsd:element default="CICS EXAMPLE CATALOG APPLICATION" name="Field2">
- <xsd:annotation>
- <xsd:appinfo source="WMQI_APPINFO">
  <fieldDesc fieldBiDi="false" fieldHidden="false" fieldPosition="2" fieldProtected="true"
        messageSetDefaultRep="IBMTerminal" />
  <dbcsDesc fieldDBCSArray="0" messageSetDefaultRep="DBCSModel" />
      </xsd:appinfo>
      </xsd:annotation>
+ <xsd:simpleType>
      </xsd:element>
+ <xsd:element default="- Main Menu" name="Field36">
+ <xsd:element default="Select an action, then press ENTER" name="Field162">
+ <xsd:element default="Action . . . ." name="Field322">
+ <xsd:element default="" name="ACTION">
+ <xsd:element default="1. List Items" name="Field341">
+ <xsd:element default="2. Order Item Number" name="Field421">
- <xsd:element default="" name="ITEM-REF">
- <xsd:annotation>
- <xsd:appinfo source="WMQI_APPINFO">
  <fieldDesc fieldBiDi="false" fieldHidden="false" fieldPosition="443" fieldProtected="false"
        messageSetDefaultRep="IBMTerminal" />
  <BMSFieldDesc fieldNamed="true" messageSetDefaultRep="BMSTerminal" />
  <dbcsDesc fieldDBCSArray="0" messageSetDefaultRep="DBCSModel" />
      </xsd:appinfo>
      </xsd:annotation>
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:maxLength value="4" />
      </xsd:restriction>
      </xsd:simpleType>
      </xsd:element>
+ <xsd:element default="" name="Field448">
+ <xsd:element default="3. Exit" name="Field501">
+ <xsd:element default="" name="MSG1">
+ <xsd:element default="F3=EXIT" name="Field1842">
+ <xsd:element default="F12=CANCEL" name="Field1853">
      </xsd:sequence>
      </xsd:group>
+ <xsd:complexType name="Exmenu">
+ <xsd:element name="Exmenu" type="Dfh0xs1_Exmenuns:Exmenu">
</xsd:schema>
```

3. Nun muss der CICS Catalog Manager gestartet werden, um zu überprüfen ob der BMS Import erfolgreich war. Hierzu starten wir im immer noch geöffnetem Hostverbindungsemulator mit dem Befehl **L CICS** das CICS und rufen mit **EGUI** die CICS Catalog Manager Anwendung auf (Abbildung 30).

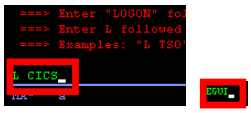


Abbildung 30

4. Wird der Hauptbildschirm des CICS Catalog Managers nicht sofort erkannt, so steht „*The screen is unrecognized.*“ (in Abbildung 31 gelb markiert) im Statusfenster des Host Emulators. Um diesen und jeden anderen Screen erneut zu laden und somit einen Refresh durchzuführen, muss auf das **Reload Screen Descriptions** Icon geklickt werden (Abbildung 31).

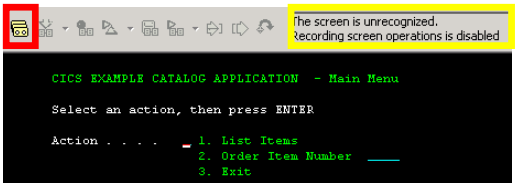


Abbildung 31

5. Danach sollte der Bildschirm des Hauptmenüs als „*Dfh0xs1.Exmenu_Exmenu*“, also korrekt, erkannt werden (Abbildung 32). Zur Sicherheit können nun auch noch die anderen Bildschirme der Applikation aufgerufen werden.

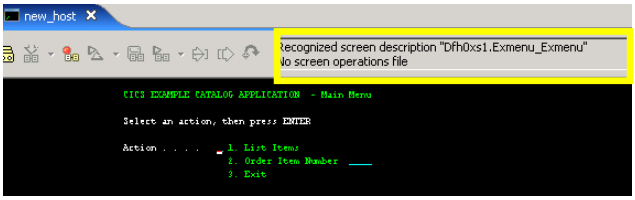


Abbildung 32

Ist alles richtig eingerichtet worden kann mit der eigentlichen Aufzeichnung des Flows begonnen werden. Für die eigentliche Flow Aufzeichnung beenden wir die Applikation mit **PF3** und starten den CICS Catalog Manager erneut mit dem Befehl **EGUI**.

8. Flow Aufzeichnen

1. Um einen neuen Flow für ein Service Flow Projekt aufzuzeichnen, klicken wir auf das **Start Recording Flow** Icon (Abbildung 33).

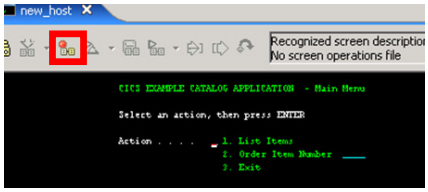


Abbildung 33

2. Das die Aufzeichnung gestartet wurde, sieht man an dem verschwundenem Start Recording Flow Icon und dem dafür erschienenem Stop Recording Flow Icons (in Abbildung 34 gelb markiert). Für unseren Flow geben wir zur Bestellung eines Artikels als Aktion **2** ein (Abbildung 34).

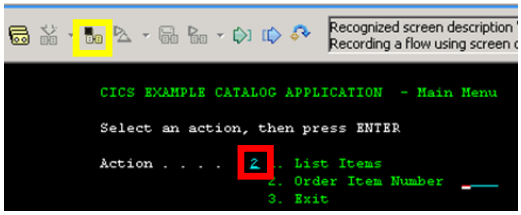


Abbildung 34

3. Damit nun die Artikelnummer des zu bestellenden Artikels als Variable erkannt wird, muss auf das **Insert Data into Screen** Icon geklickt werden (Abbildung 34). Diese und die weiteren Variablendefinitionen werden dem Message file i_OrderItem.mxsd

automatisch hinzugefügt. Die Artikelnummer wird wie zuvor schon erklärt unter dem Variablennamen ITEM-REF geführt.

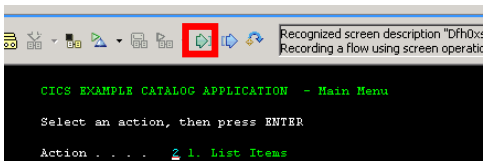


Abbildung 35

4. Bewegt man nun die Maus über das Eingabefeld der „Order Item Number“, so wird dieses blau [_ _ _ _] eingerahmt. Ist dies der Fall, so muss mit der linken Maustaste darauf geklickt werden um es als Eingabefeld für einen Input zu kennzeichnen. Anschließend geben wir in das Feld die Artikelnummer **0010** ein (Abbildung 36) und bestätigen die Eingabe mit der **Enter** Taste.

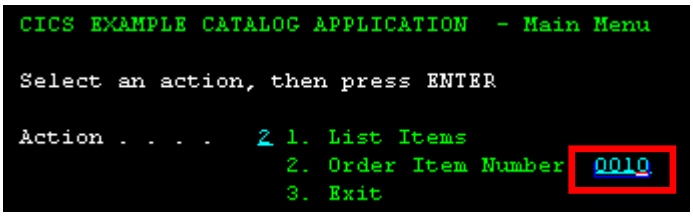


Abbildung 36

Somit gelangen wir zum nächsten Bildschirm der die detaillierte Bestellinformationen: Produktbeschreibung, Artikelpreis, Warenbestand,… liefert und drei Eingabedaten vom Benutzer benötigt: wie viel bestellt werden soll, den Benutzername und die Rechnungsstelle.

Um nun die zusätzlichen Bestellinformationen als Ausgabefelder zu übernehmen, müsse diese ähnlich der Variableneingabe definiert werden.

5. Um ein Ausgabefeld vom Bildschirm zu übernehmen klickt man auf das **Extract Data from Screen** Icon. Fährt man nun mit der Maus über das Feld der Artikelbeschreibung **Description**, so wird das Feld rot eingerahmt [ ] und wird nach einem Klick mit der linken Maustaste übernommen (Abbildung 37).

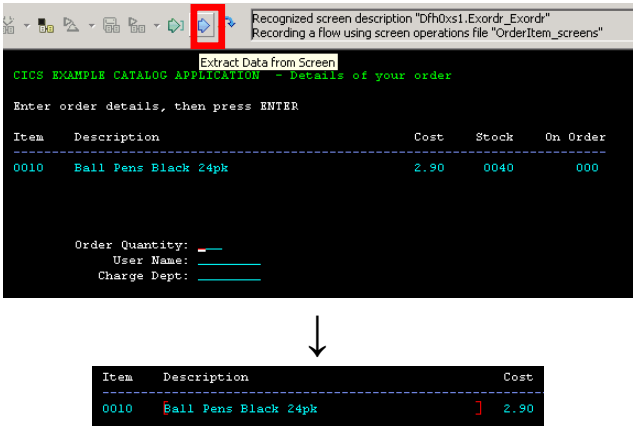


Abbildung 37

6. Schritt 6 muss für die Felder **Cost** und **Stock** wiederholt werden, bis der Bildschirm wie in Abbildung 38 dargestellt ausschaut.

Dabei ist zu beachten, dass vor jeder Felddefinition das Extract Data from Screen Icon geklickt werden muss!

Abbildung 38

Nachdem die Ausgabefelder eingelesen sind, müssen noch die drei Eingabefelder für die Variablen definiert werden.

7. Damit auch diese Felder erkannt werden, muss wieder auf das **Insert Data into Screen** Icon geklickt werden (Abbildung 39).

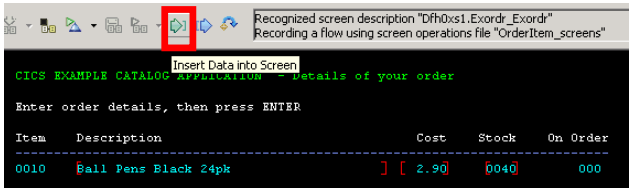


Abbildung 39

8. Bewegt man nun die Maus über das Eingabefeld „*Order Quantity*“, so wird dieses wieder blau eingerahmt **[_ _ _ _]**. Ist das der Fall, so muss mit der linken Maustaste darauf geklickt werden um es als Eingabefeld für Variablen zu kennzeichnen. Wir füllen das Feld für die Anzahl mit **001** (Abbildung 40).

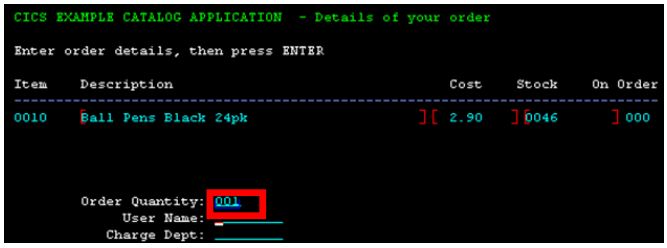


Abbildung 40

9. Schritt 8 muss für die Felder **User Name** und **Charge Dept** wiederholt werden. Wir füllen diese Felder mit den Werten **testuser** und **testdept**. Dabei ist zu beachten, dass auch hier wieder vor jeder Felddefinition das Insert Data into Screen Icon geklickt werden muss! Ist dieser Vorgang abgeschlossen und sieht der Bildschirm wie in Abbildung 41 dargestellt aus, so kann mit der **Enter** Taste die Bestellung abgeschlossen werden.

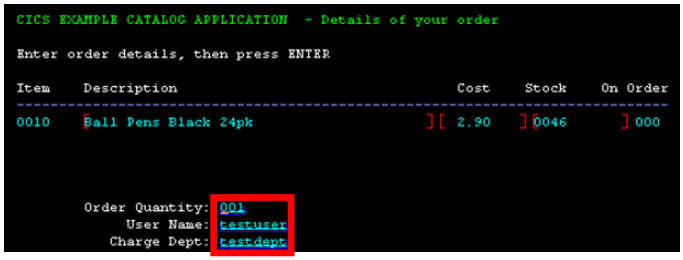


Abbildung 41

10. Man gelangt nun zurück zum Hauptmenü Bildschirm des CICS Catalog Managers, in welchem im unteren Bereich eine Statusmeldung über die soeben abgeschlossene Bestellung erscheint. Da in diesem Beispiel alle Eingaben gültige Werte besaßen und der gewünschte Artikel in ausreichender Anzahl verfügbar war, erscheint hier die Meldung: „ORDER SUCESSFULLY PLACED." Um auch diese Statusmeldung in unseren Flow aufzunehmen, markieren wir auch dieses Feld als Bildschirmausgabe. Wir verfahren wie unter Schritt 5 beschrieben (Abbildung 42).

Abbildung 42

11. Da nun das Ende unseres kleinen Flows erreicht ist, stoppen wir die Aufzeichnung durch einen Klick auf das **Stop Flow** Icon, welches an Stelle des Start Recording Flow Icons erschienen ist (Abbildung 43).

Abbildung 43

12. Um die Aufzeichnung endgültig abzuschließen muss der soeben erstellte Flow mit einem Klick auf das **Save Flow** Icon abgespeichert werden (Abbildung 44).

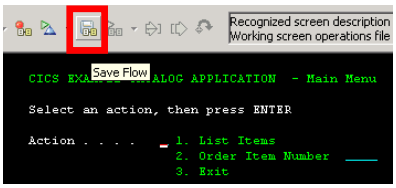


Abbildung 44

Der neue Service Flow ist hiermit aufgezeichnet und kann nun weiterverarbeitet werden. Für die zukünftigen Schritte kann der Hostverbindungsemulator weiterhin verwendet werden. Wir verlassen lediglich den CICS Catalog Manager mit **PF3**.

Schaut man nun in den EST Project Explorer so erkennt man einige neu erstellte Dateien. So wurden z.B. Operations und Mappings angelegt, die mit dem eben erstellten Service Flow verbunden sind. Aber das Hauptaugenmerk sollte hier nur auf die OrderItem.seqflow Datei gelegt werden, da diese den Service Flow grafisch darstellt. Wir öffnen sie durch einen Doppelklick darauf (Abbildung 45).

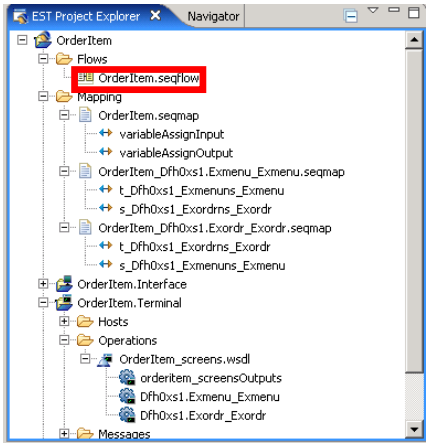


Abbildung 45

Nachdem die Datei geöffnet wurde, erkennt man zwischen dem Receive node i_OrderItem und dem Reply node o_OrderItem den eigenlichen Ablauf unseres Flows.

Der Receive node ist quasi der Startpunkt des Service Flows und der Reply node ist der Endpunkt (Abbildung 46).

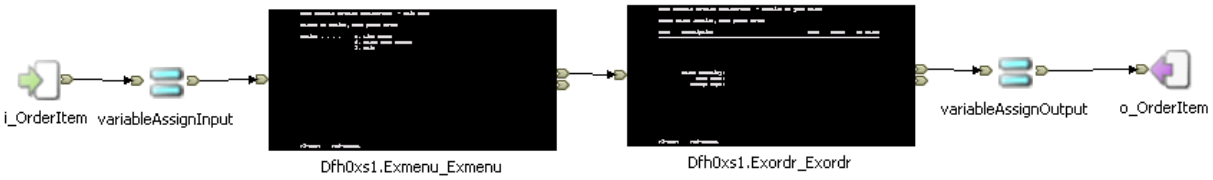


Abbildung 46

Dazwischen werden an den beiden Assign nodes variableAssignInput und variableAssignOutput Variablen – also Daten – an die Anwendung übergeben, bzw. wieder von ihr übernommen. Zwischen diesen beiden Assign nodes liegen die zwei Invoke nodes Dfh0xs1.EXmenu_Exmenu und Dfh0xs1.Exorder_Exorder, die die für die Bestellung im CICS Catalog Manager nötigen Terminal Operationen auslösen.

Expandiert man nun die sogenannte Palette, so kann der Service Flow ohne großen Aufwand bearbeitet werden (Abbildung 47). Es können Bedingungen, logische Bausteine, Schleifen oder weitere Service Flows eingefügt werden. Die gesamte Logik des Flows kann durch Hinzufügen, Entfernen oder Umverlegen von Verbindungen[3] (also diese Pfeile zwischen den jeweiligen Knoten) verändert werden.

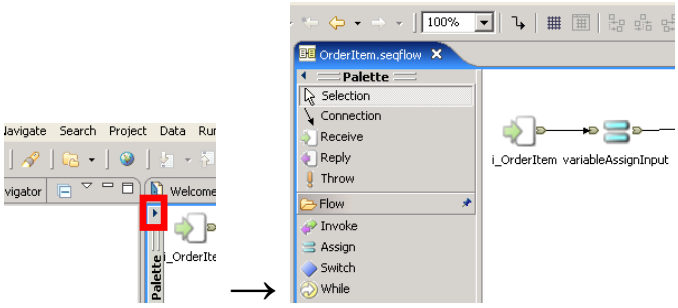


Abbildung 47

[3] sogenannte Connections

Um die Komplexität dieses kleinen Beispiels nicht unnötig zu steigern, belassen wir unseren Flow so und gehen gleich über zur Generierung des Runtime Codes.

9. Erstellen und Bearbeiten des Generation Properties Files

Bevor mit der Generierung des Runtime Codes begonnen werden kann, muss ein sogenanntes Generation Properties File existieren, bzw. wie in unserem Fall initial angelegt werden. Hierfür wird der sogenannte „*New Generation Properties*“ Wizard verwendet. Solch eine Datei enthält verschiedenste Parameter, die für die Generierung des Runtime Codes notwendig sind und dient quasi als Eingabedatei für die im Anschluss noch nötige Erstellung des Runtime Codes. Somit ist es möglich, ein solches property file einmalig zu erstellen und für unterschiedliche Projekte zu benutzen.

1. Um den New Generation Properties Wizard zu starten klicken wir im EST Projekt Explorer mit der rechten Maustaste auf die OrderItem.seqflow Datei. Und wählen **New Generation Properties File** aus (Abbildung 48).

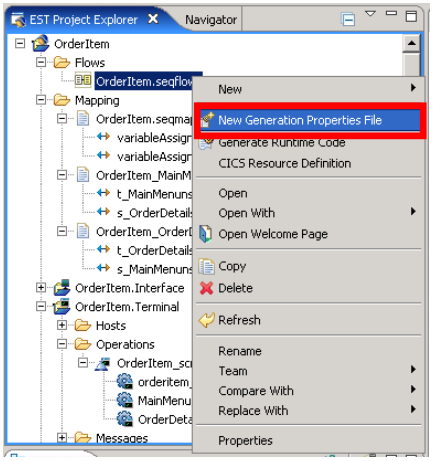


Abbildung 48

2. Im ersten Schritt des nun geöffnetem Wizards muss der Dateiname für das property file eingegeben werden. Wir wählen **OrderItemGenProps.wsdl** (Abbildung 49). Im Falle, dass mit mehreren Flows gearbeitet wird, besteht die Möglichkeit, für jeden ganz individuell solch ein propertiy file anzulegen, bzw. zuzuordnen. Da wir hier nur mit einem Flow arbeiten, sollte standardmäßig das richtige Projekt (OrderItem) und der richtige Flow (OrderItem.seqflow) ausgewählt sein. Wir können nun mit **Next** zum zweiten Schritt wechseln.

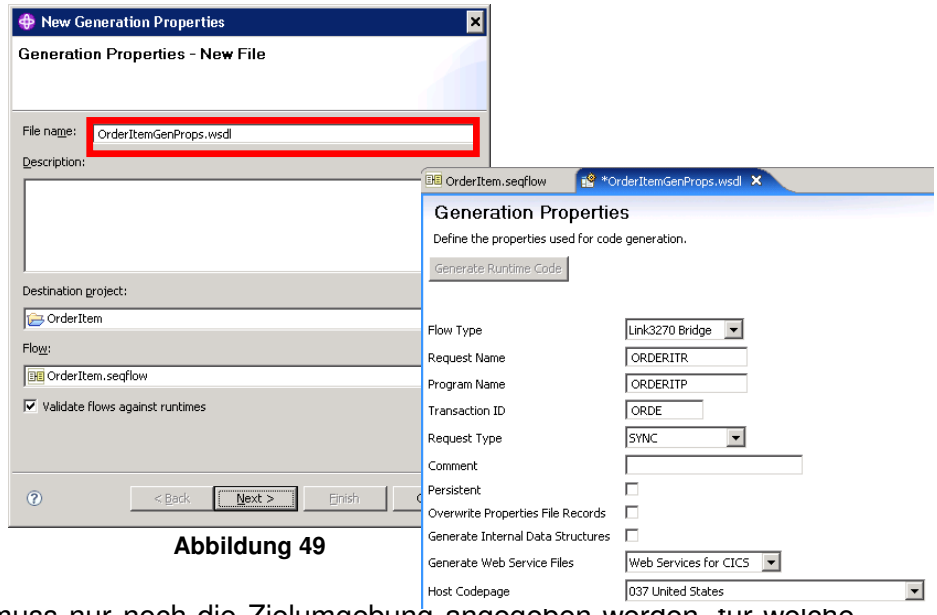


Abbildung 49

3. Im zweiten Schritt muss nur noch die Zielumgebung angegeben werden, für welche das properties file erstellt werden soll. Da wir einen Service Flow erstellen wollen, wählen wir hier **CICS Service Flow Runtime** und schließen den Wizard mit **Finish** (Abbildung 50).

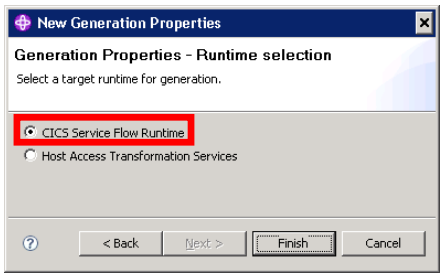


Abbildung 50

Die Datei OrderItemGenProps.wsdl wird nun erstellt und anschließend geöffnet.

4. In der geöffneten Datei OrderItemGenProps.wsdl verändern wir die folgenden Werte zu:

Achtung alle Werte sind case sensitive!

Flow Type:	Link3270Bridge
Request Name:	ORDERITR

Program Name:	ORDERITP
Transaction ID:	ORDE
Generate Web Service Files:	Web Services for CICS
Initial PFKey:	CLEAR
Startup Transaction Data:	EGUI
End Point URI:	http://139.18.4.34:3601/exampleApp/ws
Local URI:	/exampleApp/ws (autom. eingetragen)
WSBIND File Name:	orderitem
WSDL File Name:	orderitem
WSDL HFS File Path:	/u/FSTEFA2/wsdl

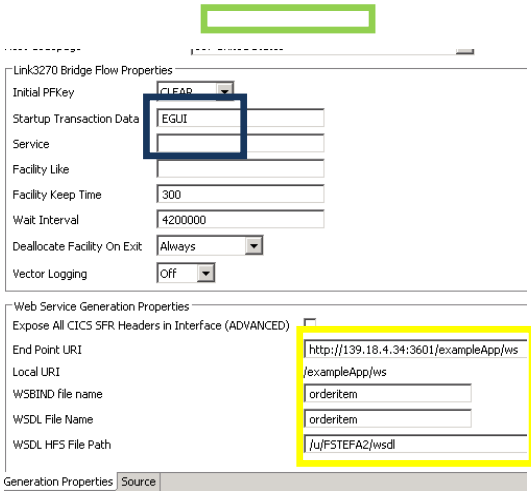


Abbildung 51

Somit ist das Generation Properties file so ausgefüllt, wie in Abbildung 51 dargestellt. Anschließend müssen die Veränderungen noch mit einem Druck auf die Tastenkombination **Strg + S** gespeichert werden.

5. Wenn man anschließend wieder in den oberen Teil der Eigenschaften scrollt, so bemerkt man, dass der Generate Runtime Code Button nun auswählbar ist. Um den Runtime Code zu erzeugen drücken wir nun diesen **Generate Runtime Code** Button (Abbildung 52).

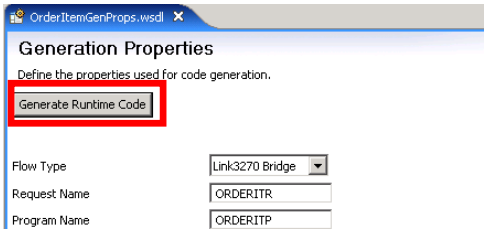


Abbildung 52

10. Generieren und Deployen des Runtime Codes

Der Wizard, welcher sich eben geöffnet hat, wird als „*Generate runtime code*“ Wizard bezeichnet. Voraussetzung für einen erfolgreichen Deploy ist, dass die benötigten Ressourcen wie unter 6. beschrieben angelegt wurden und eine Hostverbindung besteht.

1. Im ersten Schritt muss die Checkbox **Deploy to remote target location** selektiert werden, da sonst der Code nur lokal erzeugt werden würde. Weiterhin geben wir ein:

Job Control user ID:	FSTEFA2
Job Control account:	FSTEFA2
Deployment libraries HLQ:	FSTEFA2

Nachdem der screen nun so ausgefüllt ist wie in Abbildung 53 dargestellt, kann mit **Next** in den zweiten Schritt gewechselt werden.

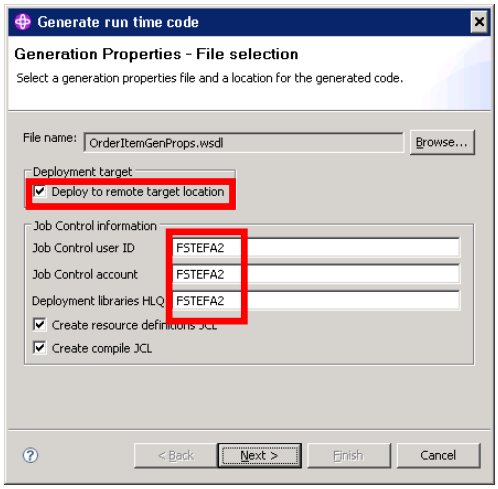


Abbildung 53

2. Da wir den neuen Code nicht zu einem anderen lokalen Projekt hinzufügen wollen, muss im unteren Bereich des zweiten Schrittes **Deploy to remote location** ausgewählt und einmalig folgende Daten eingegeben werden:

Remote location adress:	139.18.4.34
Remote source code location:	FSTEFA2.USER.SRCLIB
Remote copy member location:	FSTEFA2.USER.COPYLIB
Remote JCL location:	FSTEFA2.USER.JCLLIB
Remote location address:	139.18.4.34
Remote WebService BIND file location:	/u/FSTEFA2/wsbind
Remote WSDL file location:	/u/FSTEFA2/wsdl

Zu beachten ist hier wierde, dass die USS Deployment Informationen case sensitive sind.

Es muss also wirklich /u/FSTEFA2/… eingegeben werden. Existiert solch ein Verzeichnis nicht, so erscheint im oberen Bereich dieses Fensters eine Warnung. Nachdem die WSDL file location eingegeben wurde, muss der Cursor noch in ein anderes Feld gebracht werden (beispielsweise zur WSBIND file location), damit der Wizard registriert, dass die Eingabe beendet ist. Sind alle Daten richtig eingegeben, so wird der Button Finish auswählbar. Wir schließen den Wizard und starten die

Generierung und das Deployment des Codes durch einen Klick auf **Finish** (Abbildung 54). Nachdem ein Deploy erfolgreich war, werden die Generierungseigenschaften für das aktuelle und alle weiteren Projekte verwendet, bis der Workspace gewechselt wird.

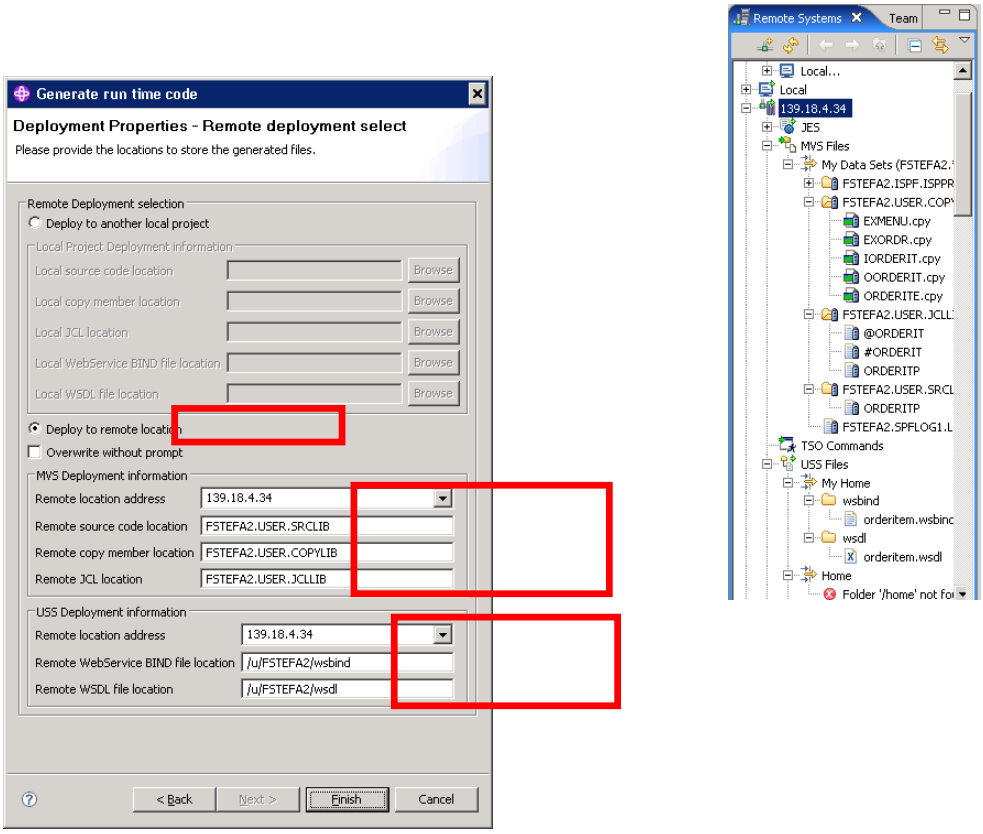


Abbildung 54

3. Nun erfolgt die lokale Code Generierung und das Deployment auf den Host. Verläuft alles ohne Komplikationen, so erscheint nach kurzer Wartezeit die Nachricht „*Generation completed sucessfully*.“. Der Runtime Code wurde ordnungsgemäß lokal und auf dem Host erstellt. Dieses Fenster kann mit **OK** geschlossen werden.

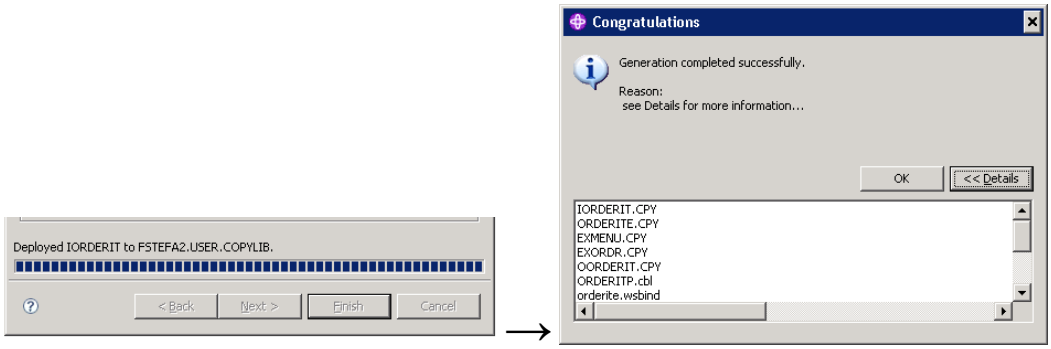


Abbildung 55

11. Überprüfen und Bearbeiten des erstellten Runtime Codes

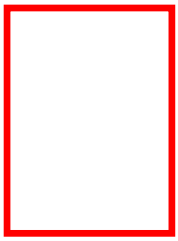
Um zu überprüfen ob der Runtime Code auch richtig auf den Host kopiert wurde, muss wieder in die „*z/OS Projects*“ Perspektive gewechselt werden:

Window → Open Perspective → Other → z/OS Projects (default)

Man sieht nun, dass wie in Abbildung 56 zu sehen ist:

- fünf copybooks,
- drei JCL Dateien
- und eine cobol Datei im MVS,
- sowie jeweils eine wsbind und wsdl Dateim im USS

angelegt wurden.



In diesen Dateien steckt der gesamte Service Flow. Ist dieser komplexerer Natur, so können es durchaus mehr copybooks und cobol Dateien sein. Es sind jedoch immer die gleichen JCL Dateien, die auch immer den gleichen Aufbau besitzen. Also eine beginnt immer mit @, die andere mit # und die dritte trägt den Programmnamen.

Abbildung 56

Erstellt werden diese Dateien nach Vorgaben, die die unter 6. erwähnten JCL Templates definieren. Hier wäre es auch problemlos möglich Änderungen vorzunehmen, sodass die nachfolgenden Änderungsschritte nicht mehr nötig wären.

Diese drei JCLs müssen nun noch manuell mit wenigen Handgriffen an das jeweils verwendete System angepasst werden.

1. **Ändern der @ORDERIT.jcl Datei.** Diese Datei muss mit einem Doppelklick im Remote System Explorer geöffnet werden. Danach sind die folgenden drei gelb hervorgehobenen Teile zu ändern.

```
//FSTEFA2C JOB (FSTEFA2),'FSTEFA2',
// CLASS=A,MSGCLASS=H,NOTIFY=FSTEFA2
//*************************** DFHMAXPU ********************************
//* @START_RRS_COPYRIGHT@
//* VERSION: 0
//*
//* Licensed Materials - Property of IBM
//*
//* 5655-M15
//*
//* (C) Copyright IBM Corp. 2000, 2005
//* @END_RRS_COPYRIGHT@
//*********************************************************************
//*********************************************************************
//* RUN DFHMAMUP (CICS SFR PROPERTIES FILE UPDATE PROGRAM)
//*
//*********************************************************************
//MAMUP    EXEC PGM=DFHMAMUP
//STEPLIB DD DSN='qual.SCIZLOAD',DISP=SHR
//*********************************************************************
//*     THE OUTPUT WILL GO TO SYSPRINT
//*********************************************************************
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//DFHMAMPF DD DSN='qual.DFHMAMPF',DISP=SHR
//*********************************************************************
//*                                                                   *
//*   SYSIN CONTROL CARD FORMATS                                      *
//*                                                                   *
//**  CARD=TYPE  ***************************************************
//*                                                                   *
//*  TYPE (R-REQUEST; 0-NAVIGATOR; 1-DPL; 2-MQPUT; 3-FEPI; 4-MQGET;   *
//*                5-LINK3270)                                        *
//*                                                                   *
//**  CARD=NAME  ***************************************************
//*                                                                   *
//*  NAME (REQUEST NAME/COMMAND NAME)                                 *
//*                                                                   *
```

… DSN='DFH310.SFFPROD.SCIZLOAD',DISP=SHR

… DSN='DFH310.SFFPROD.DFHMAMPF',DISP=SHR

```
//**  PARMS TYPE=R  ***************************************************
//*                                                                    *
//*  PARM01     : REQUEST TYPE (0 - Asynchronous;                      *
//*                             (1 - Synchronous;                      *
//*                             (2 - Synchronous Rollback)             *
//*  PARM02     : NAVIGATOR/INITIAL PROGRAM NAME                       *
//*  PARM03     : NAVIGATOR/INITIAL TRANSACTION ID                     *
//*  PARM04     : PERSISTENCE INDICATOR (0 - Nonpersistent;            *
//*                                      1 - Persistent)               *
//*  PARM05     : XML PARSER PROGRAM NAME                              *
//*  PARM06     : DEPLOYMENT INDICATOR (1 - Simple sequence;           *
//*                                     2 - Complex sequence)          *
//*  PARM07     : INITIAL PROGRAM TYPE (0 - Navigator;                 *
//*                                     1 - DPL;                       *
//*                                     2 - MQPUT;                     *
//*                                     3 - FEPI;                      *
//*                                     4 - MQGET;                     *
//*                                     5 - Link3270)                  *
//*                                                                    *
//**  PARMS TYPE=0  ***************************************************
//*                                                                    *
//*  (RESERVED)                                                        *
//*                                                                    *
//**  PARMS TYPE=1  ***************************************************
//*                                                                    *
//*  PARM01     : DPL PROGRAM                                          *
//*  PARM02     : DPL SYSID                                            *
//*  PARM03     : DPL MIRROR TRANSACTION                               *
//*  PARM04     : DPL SYNCONRETURN (Y/N)                               *
//*                                                                    *
//**  PARMS TYPE=2  ***************************************************
//*                                                                    *
//*  PARM01     : MQ MSGTYPE  (1-REQUEST; 2-REPLY; 8-DATAGRAM)         *
//*  PARM02     : MQ REQUEST QNAME                                     *
//*  PARM03     : MQ REPLYTOQ                                          *
//*  PARM04     : MQ REPLYTOQMGR                                       *
//*                                                                    *
//**  PARMS TYPE=3  ***************************************************
//*                                                                    *
//*  PARM01     : FEPI POOL NAME                                       *
//*  PARM02     : FEPI TARGET NAME                                     *
//*  PARM03     : FEPI (BE) TIMEOUT VALUE                              *
//*  PARM04     : FEPI EXIT ACTION (R-RELEASE; F-FORCE; P-PASS;        *
//*                                 H-HOLD; A-LEAVE ASSIGNED)          *
//*  PARM05     : FEPI SIGNON PROGRAM NAME  (NOT CURRENTLY SUPPORTED) *
//*  PARM06     : FEPI SIGNOFF PROGRAM NAME (NOT CURRENTLY SUPPORTED) *
//*  PARM07     : FEPI ROUTING PROGRAM NAME (NOT CURRENTLY SUPPORTED) *
//*  PARM08     : FEPI NON-UNIQUE USERID                               *
//*               (Y - NON-UNIQUE USERIDS IN USE.                      *
//*                    UNIQUE ASSIGNMENT REQUIRED.)                    *
//*               (N - UNIQUE USERIDS IN USE                           *
//*                    NO UNIQUE ASSIGNMENT REQUIRED.)                 *
//*                                                                    *
//**  PARMS TYPE=4  ***************************************************
//*                                                                    *
//*  PARM01     : MQ WAIT LIMIT (SECONDS)                              *
//*  PARM02     : MQ REPLYTOQ                                          *
//*  PARM03     : MQ REPLYTOQMGR                                       *
//*                                                                    *
//**  PARMS TYPE=5  ***************************************************
//*                                                                    *
//*  PARM01     : LINK3270 SERVICE NAME                                *
//*  PARM02     : LINK3270 FACILITYLIKE                                *
//*  PARM03     : LINK3270 MAX. FACILITY KEEPTIME (seconds)            *
//*  PARM04     : LINK3270 GETWAITINTERVAL (milli-seconds)             *
//*  PARM05     : LINK3270 DEALLOCATE FACILITY ON EXIT                 *
//*                (0-NO; 1-ALWAYS; 2-IF SUCCESSFUL;                   *
//*                 3-IF UNSUCCESSFUL)                                 *
//*  PARM06     : LINK3270 NON-UNIQUE USERID ASSIGNMENT                *
//*                (Y - NON-UNIQUE USERIDS IN USE.                     *
//*                     UNIQUE ASSIGNMENT REQUIRED.)                   *
//*                (N - UNIQUE USERIDS IN USE                          *
//*                     NO UNIQUE ASSIGNMENT REQUIRED.)                *
//*  PARM07     : LINK3270 AOR ROUTING ALLOWED/IN USE                  *
//*                (Y - AOR ROUTING ALLOWED/IN USE)                    *
//*                (N - AOR ROUTING NOT ALLOWED/IN USE)                *
//*  PARM08     : LINK3270 VECTOR LOGGING  (0 - OFF)                   *
```

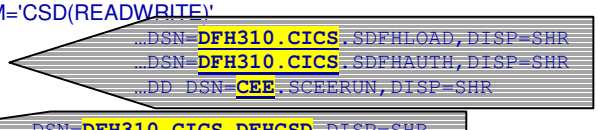
```
//*                          (1 - FULL)              *
//*                          (2 - TRACE)               *
//*                                                *
//*****************************************************************
//SYSIN   DD *
MODE=SAFE
TYPE=5
NAME=ORDERITP
PARM01=
PARM02=
PARM03=300
PARM04=4200000
PARM05=1
PARM06=
PARM07=Y
PARM08=0
PARMXX
TYPE=R
NAME=ORDERITR
PARM01=1
PARM02=ORDERITP
PARM03=ORDE
PARM04=0
PARM05=
PARM06=1
PARM07=5
PARMXX
/*
//
****************** TRAILER: DFHMAXPU ********************************
```

Anschließend muss die Datei @ORDERIT.jcl noch mit der Tastenkombination **Strg + S** gespeichert werden. Danach kann sie mit einem Klick auf 

geschlossen werden.

2. **Ändern der #ORDERIT.jcl Datei.** Auch diese Datei muss mit einem Doppelklick im Remote System Explorer geöffnet werden. Danach sind die folgenden gelb hervorgehobenen Teile zu ändern.

```
//FSTEFA2C JOB (FSTEFA2),'FSTEFA2',
// CLASS=A,MSGCLASS=X,NOTIFY=FSTEFA2
//************************** DFHMAXRD *********************************
//* @START_RRS_COPYRIGHT@
//* VERSION: 0
//*
//* Licensed Materials - Property of IBM
//*
//* 5655-M15
//*
//* (C) Copyright IBM Corp. 2000, 2005
//* @END_RRS_COPYRIGHT@
//*****************************************************************
//*****************************************************************
//* RDO RESOURCE DEFINITIONS
//*****************************************************************
//DEFINE   EXEC PGM=DFHCSDUP,REGION=1024K,PARM='CSD(READWRITE)'
//STEPLIB  DD DSN=hlqcics.SDFHLOAD,DISP=SHR
//         DD DSN=hlqcics.SDFHAUTH,DISP=SHR
//         DD DSN=hlqcee.SCEERUN,DISP=SHR
//* following line changed by APAR PK17446
//DFHCSD   DD DSN=csdname,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
************************** DFHMAXRP **********************************
* @START_RRS_COPYRIGHT@
* VERSION: 0
*
* Licensed Materials - Property of IBM
*
* 5655-M15
```



```
*
* (C) Copyright IBM Corp. 2000, 2005
* @END_RRS_COPYRIGHT@
********************************************************************
 DEFINE PROGRAM(ORDERITP) GROUP(CICSSFRG)          ... GROUP (SFRG)
 DESCRIPTION('')
       LANGUAGE(LE370) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(NO) DATALOCATION(ANY)
       EXECKEY(USER) CONCURRENCY(QUASIRENT) DYNAMIC(NO)
       EXECUTIONSET(FULLAPI) JVM(NO)

****************** TRAILER: DFHMAXRP **********************************

************************** DFHMAXRT **********************************
* @START_RRS_COPYRIGHT@
* VERSION: 0
*
* Licensed Materials - Property of IBM
*
* 5655-M15
*
* (C) Copyright IBM Corp. 2000, 2005
* @END_RRS_COPYRIGHT@
********************************************************************
 DEFINE TRANSACTION(ORDE) GROUP(CICSSFRG)          ... GROUP (SFRG)
       DESCRIPTION('')
       PROGRAM(ORDERITP) TWASIZE(0) PROFILE(DFHCICST)
       STATUS(ENABLED) TASKDATALOC(ANY) TASKDATAKEY(USER)
       RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
       ROUTABLE(NO) PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO)
       RESTART(NO) SPURGE(YES) TPURGE(NO) DUMP(YES) TRACE(YES)
       CONFDATA(NO) ACTION(BACKOUT) WAIT(YES) WAITTIME(0,0,0)
       RESSEC(NO) CMDSEC(NO) STORAGECLEAR(NO)

****************** TRAILER: DFHMAXRT **********************************

************************** DFHMAXRG **********************************
* @START_RRS_COPYRIGHT@
* VERSION: 0
*
* Licensed Materials - Property of IBM
*
* 5655-M15
*
* (C) Copyright IBM Corp. 2000, 2005
* @END_RRS_COPYRIGHT@
********************************************************************
* ADD GROUP(CICSSFRG) LIST(CICSSFRL)          ... GROUP (SFRG) LIST (CICSSFRL)

****************** TRAILER: DFHMAXRG **********************************

************************** DFHMAXRR **********************************
* @START_RRS_COPYRIGHT@
* VERSION: 0
*
* Licensed Materials - Property of IBM
*
* 5655-M15
*
* (C) Copyright IBM Corp. 2006
* @END_RRS_COPYRIGHT@
********************************************************************
 DEFINE PROCESSTYPE(ORDERITR) GROUP(CICSSFRG)      ... GROUP (SFRG)
       DESCRIPTION('')
       STATUS(ENABLED)
       FILE(BTS) AUDITLEVEL(OFF)

****************** TRAILER: DFHMAXRR **********************************

/*
//
//****************** TRAILER: DFHMAXRD **********************************
```

Anschließend muss die Datei #ORDERIT.jcl noch mit der Tastenkombination **Strg + S** gespeichert werden. Danach kann sie geschlossen werden.

3. **Ändern der ORDERITP.jcl Datei.** Auch diese Datei muss mit einem Doppelklick im Remote System Explorer geöffnet werden. Danach sind die folgenden gelb hervorgehobenen Teile zu verändern.

```
//FSTEFA2C JOB (FSTEFA2),'FSTEFA2',
// CLASS=A,MSGCLASS=H,NOTIFY=FSTEFA2
//************************** DFHMAXCJ *********************************
//* @START_RRS_COPYRIGHT@
//* VERSION: 0
//*
//* Licensed Materials - Property of IBM
//*
//* 5655-M15
//* (c) Copyright IBM Corp. 2000, 2005
//* @END_RRS_COPYRIGHT@
//*********************************************************************
//*
//   JCLLIB ORDER=(qual.SCIZSAMP)
//*
//* CURRENTLY SET FOR: ORDERITP
//*
//COMPILE EXEC DFHMAXCP,
//         COPY1='FSTEFA2.USER.COPYLIB',
//         COPY2='qual.SCIZMAC',
//         COPY3='hlqcics.SDFHCOB',
//         LINK2='qual.SCIZLOAD',
//         OBJLIB='qual.SCIZLOAD',
//         PROGRAM='ORDERITP',
//         LINK='FSTEFA2.USER.LINKLIB',
//         SOURCE='FSTEFA2.USER.SRCLIB(ORDERITP)'
//LINKSTEP.SYSIN    DD *
  INCLUDE OBJLIB(DFHMAF)
/*
//*MQ   DD DISP=SHR,
//*MQ    DSN=qual.SCIZSAMP(DFHMAIN0)
//       DD *
  NAME ORDERITP(R)
/*
//
//****************** TRAILER: DFHMAXCJ *********************************
```

… ORDER=(DFH310.SFFPROD.SCIZSAMP)

…COPY2='DFH310.SFFPROD.SCIZMAC',
…COPY3='DFH310.CICS.SDFHCOB',
…LINK2='DFH310.SFFPROD.SCIZLOAD',
…OBJLIB='DFH310.SFFPROD.SCIZLOAD',

… LINK='DSN810.RUNLIB.LOAD',

Anschließend muss die Datei ORDERITP.jcl noch mit der Tastenkombination **Strg + S** gespeichert werden. Danach kann sie geschlossen werden.

12. Installation des Webservices

1. Nun sind die Änderungen an den JCLs komplett und wir können die Jobs an das System abschicken. Hierzu öffnen wir einen neuen Host Verbindungsemulator durch einen Rechtsklick auf die Verbindung 139.18.4.34 und melden uns im **TSO** mit:

   Benutzername: FSTEFA2
   Passwort: uni4you an.

2. Nach erfolgreichem Anmelden, lassen wir uns mit dem unter **3.4** erreichbarem Data Set List Utility den Inhalt von unserem Dataset **FSTEFA2.USER.JCLLIB** anzeigen (Abbildung 57).

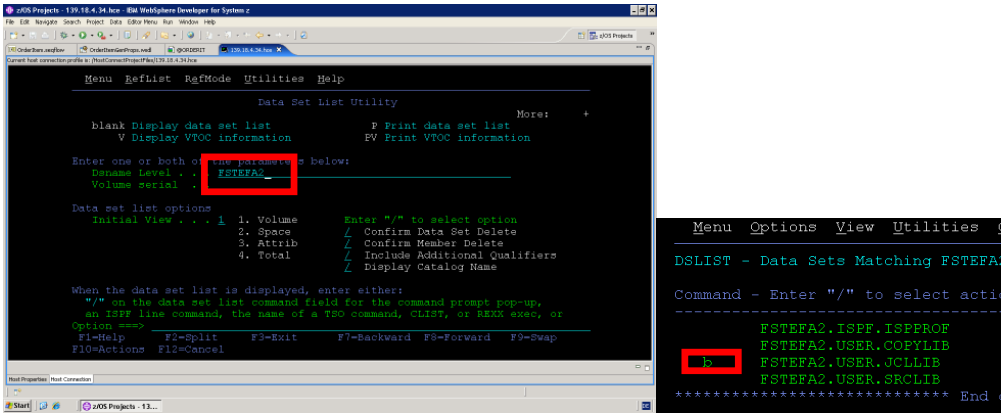


Abbildung 57

3. Wir sehen im Dataset **FSTEFA2.USER.JCLLIB** die vom Service Flow Modeler erzeugten JCL Dateien: #ORDERIT, @ORDERIT und ORDERITP. Zum Abschicken an das System geben wir vor jeder Datei den Befehl **sub** ein und drücken danach die **Enter** Taste (Abbildung 58).

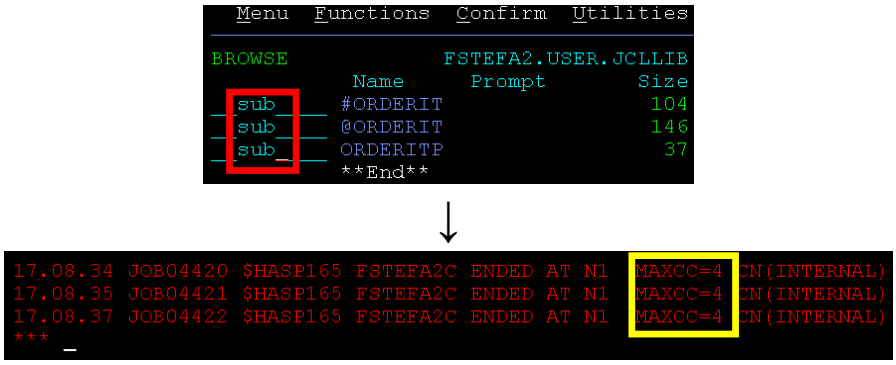


Abbildung 58

Wurden alle Änderungen an den JCL Dateien richtig gemacht und alle copybooks und das cobol Programm richtig deployed, so liefern alle drei Jobs eine Fehler kleiner gleich 4 zurück. Die Arbeit im TSO ist nun beendet und wir können uns mit **logoff** abmelden und diesen Hostverbindungsemulator schließen. Wir arbeiten von jetzt ab nur noch im noch offenen Host Emulator - in der CICS Umgebung.

4. Im CICS Hostverbindungsemulator angekommen lassen wir uns als erstes die Gruppe SFRG mit dem Befehl **CEDA DISPLAY GROUP(SFRG)** anzeigen (Abbildung 59). Wir sehen, dass durch die JCLs das Programm ORDERITP, die Transaction ORDE und der Prozesstype ORDERITR angelegt wurden. Die Pipeline SFPIPEG wurde bereits zuvor manuell angelegt und definiert folgendes:
    - einen Serviceprovider, der unter /usr/lpp/cicsts/cicsts31/samples/pipelines/basicsoap11provider.xml zu finden ist,
    - ein SHelf Verzeichnis, in welches der Webservice installiert wird (hier: /var/cicsts)
    - das Webservice Verzeichnis, in welchem die Datei orderitem.wsbind abgelegt wurde (hier: /u/FSTEFA2/wsbind)

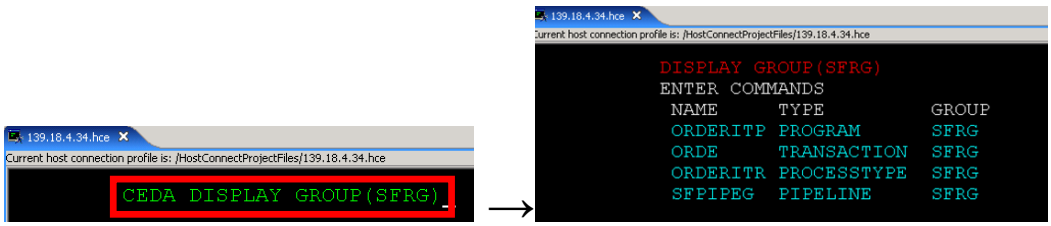


Abbildung 59

5. Um nun das Programm ORDERITP und die Transaction ORDE zu installieren geben wir hinter diese beiden ein **i** wie install ein und drücken danach die **Enter** Taste (Abbildung 60). Anschließend erscheint hinter jedem diesen beiden ein INSTALL SUCCESSFUL. Der Webservice ist nun ordnungsgemäß installiert. Der Prozesstype ORDERITR wurde bereits während der Übermittlung der #ORDERIT.jcl Datei an das System installiert.

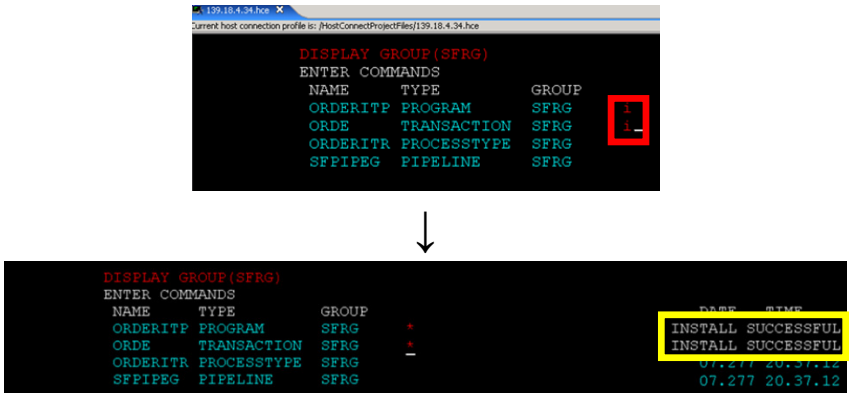


Abbildung 60

6. Doch bevor der Webservice vom CICS verwendet werden kann, muss er diesem bekannt gemacht werden. Normalerweise geschieht das beim der Installation einer Pipeline. Doch da in unserem Fall diese bereits installiert war, müssen wir noch zu allerletzt einen Pipeline Scan durchführen. Dies muss im Übrigen immer getan werden, wenn ein neuer Webservice in das Webservice Verzeichnis installiert wird. Wir benutzen für den Pipeline Scan den Befehl:

   CEMT PERFORM PIPELINE(SFPIPEB) SCAN.

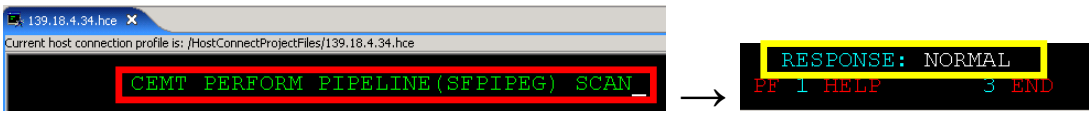


Abbildung 61

   Ist dieser erfolgreich abgeschlossen, so sollte im linken unteren Bereich die Nachricht „*RESPONSE: NORMAL*“ erscheinen(Abbildung 61).

Somit prüft die Pipeline ihr Arbeitsverzeichnis, also /u/FSTEFA2/wsbind auf neue wsbind Dateien. In unserem Fall findet die Pipeline die orderitem.wsbind Datei und kopiert diese in das SHelf Verzeichnis **/var/cicsts**/CICS/PIPELINE/SFPIPEB.

Die Installation des Webservices ist nun abgeschlossen und der neue Webservice kann nun getestet werden.

13. Testen des Webservices mit dem Web Services Explorer

In diesem letzten Abschnitt wollen wir nun den Webservice im Web Services Explorer ausführen. Dazu verwenden wir die lokalen Informationen der orderitem.wsdl Datei, die bei der Generierung des Runtime Codes erstellt wurde.

1. Um den Web Service Explorer zu starten muss mit der rechten Maustaste im EST Project Explorer auf die orderitem.wsdl Datei geklickt werden. Und anschließend

   Web Services → Test with Web Services Explorer

   ausgewählt werden (Abbildung 62).

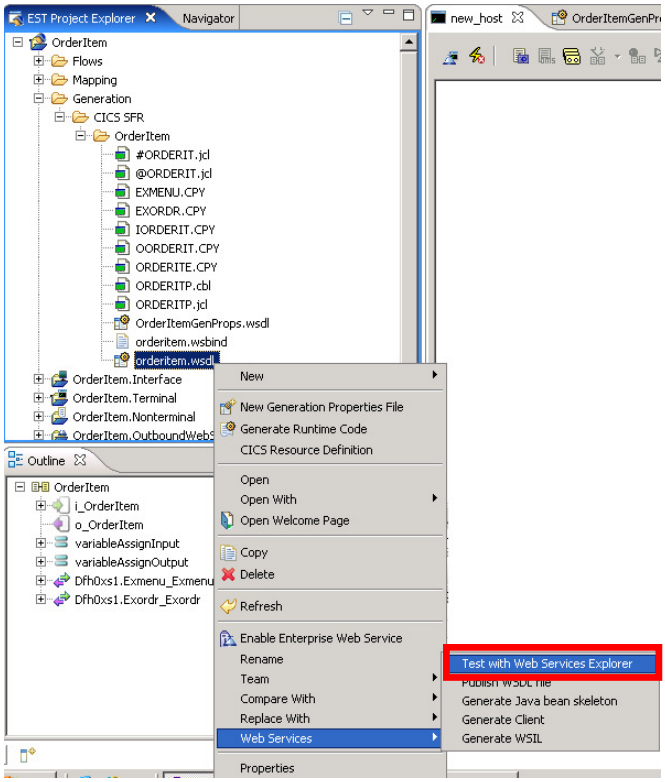


Abbildung 62

2. Nun öffnet sich normalerweise ein neues Fenster, der Web Services Explorer. Ist dies nicht der Fall und schaut der Bildschirm wie in Abbildung 63 dargestellt aus, so muss nur auf **Open file using the system editor** geklickt werden.

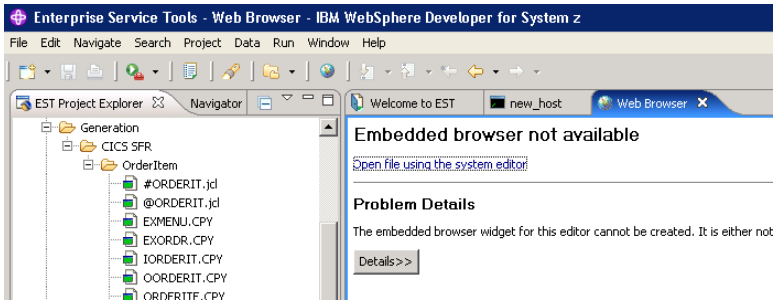


Abbildung 63

Wir sehen als initiale Statusmeldung, daß die lokale orderitem.wsdl Datei erfolgreich geöffnet werden konnte. Da diese unseren deployment Informationen entspricht, kann ohne Änderungen der Endpoints mit dem Testen begonnen werden. Dazu

expandiert man den Eintrag des Conversation Templates DFHMADPLHTTPSoapBinding (Abbildung 64). Dieses Beschreibt, wie die CICS Service Flow Runtime bei der Datenumwandlung zwischen dem Service Flow und dem Web Services Explorer vorgeht.

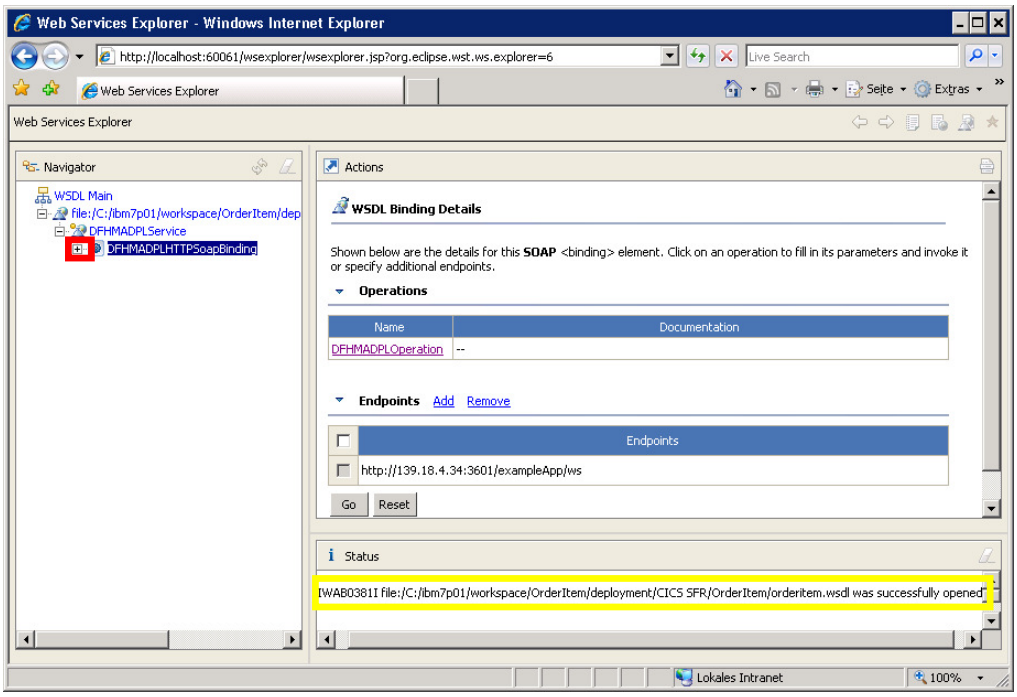


Abbildung 64

3. Nach dem Expandieren erscheint ein Eintrag namens **DFHMADPLOperation**. Wir wählen diesen mit der linken Maustaste aus. Es erscheint nun in der rechten oberen Hälfte des Web Services Explorer eine Eingabemaske für die Daten, die unser Flow als Eingaben benötigt (Abbildung 65).

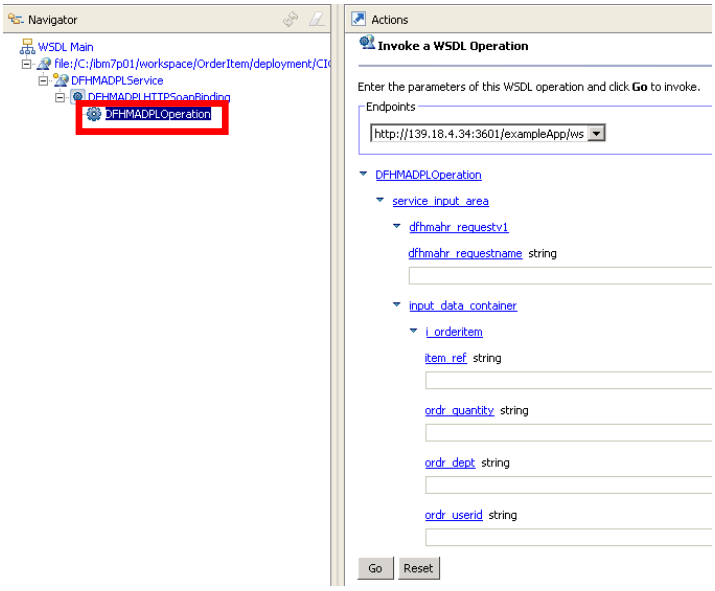


Abbildung 65

4. Wir füllen diese Eingabemaske aus wie nachfolgend beschrieben, wobei zu beachten ist, dass das Eingabefeld für den Webservice Namen case sensitive ist. Es muss also wirklich ORDERITR im Feld dfhmahr_requestname eingegeben werden.

dfhmahr_requestname:	ORDERITR
i_orderitem:	0030
ordr_quantity:	001
ordr_dept:	testdept
ordr_userid:	testuser

Danach senden wir unsere Daten mit einem Klick auf **OK** zum Webservice (Abbildung 66).

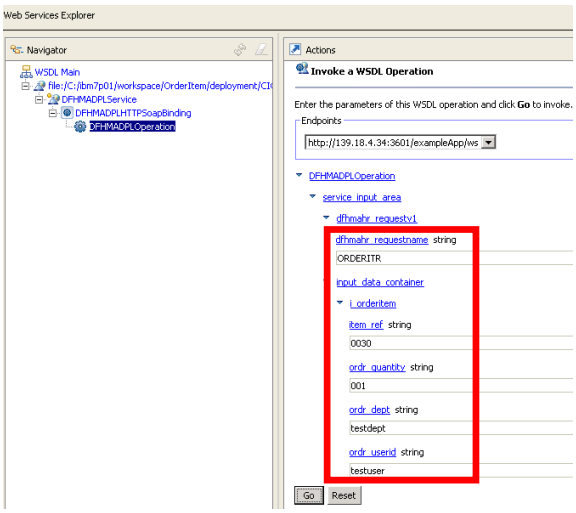


Abbildung 66

5. Im rechten unteren Bereich des Web Services Explorer erscheint nun eine Statusmeldung, was vom Webservice zurückgeschickt wurde (Abbildung 67). Wir sehen die vier von uns unter 8.5 bis 8.10 definierten Ausgabefelder.

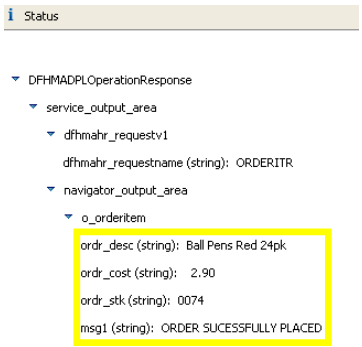


Abbildung 67

Die Statusmeldung im Feld msg1 liefert als Ergebnis: „*ORDER SUCCESSFULLY PLACED*“. Damit ist die Bestellung erfolgreich abgeschlossen und unser kleines Tutorial hiermit beendet.