TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Fakultät Informatik

Automatisierte Provisionierungsmechanismen für Laufzeitumgebungen von Legacy z/OS Anwendungen mit „IBM Cloud Provisioning and Management for z/OS“ am Beispiel der „Rechnungsschreibung“ bei DATEV eG

Bachelorarbeit im Studiengang Informatik

vorgelegt von

David Krug

Matrikelnummer 3036355

Erstgutachter:	Prof. Dr. Korbinian Riedhammer
Zweitgutachter:	Prof. Dr. Friedhelm Stappert

Hinweis: Diese Erklärung ist in alle Exemplare der Abschlussarbeit fest einzubinden. (Keine Spiralbindung)

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Vorname: Matrikel-Nr.:

Fakultät: Studiengang:

Semester:

Titel der Abschlussarbeit:

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☐ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,

☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigefügt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Ort, Datum, Unterschrift Studierende/Studierender

Kurzdarstellung

Deutsche Kurzzusammenfassung Zitattest 1 [**?**] Zitattest 2 [**?**]

Abstract

english translation of ‘kurzzusammenfassung‘

Inhaltsverzeichnis

Kapitel 1

Einleitung

‘I recently predicted the last mainframe will be unplugged on March 15, 1996‘[1], ein in der Großrechner-Welt bekannt gewordenes Zitat. Es handelt sich um eine 1993 getroffene Vorhersage, nämlich dass der letzte Mainframe, auch Großrechner genannt, am 15 März 1996 abgeschaltet werden wird. Wieso wird sich im Jahre 2020 dennoch mit dieser Technologie beschäftigt? Bevor darauf eingegangen wird, wird erläutert was ein Großrechner ist. In einem Satz ist ein Großrechner ein leistungsstarkes, zentralisiertes Serversystem. Eine genauere Beschreibung ist im Absatz **??** zu finden. In dieser Arbeit wird nur auf Mainframes aus dem Hause von IBM eingegangen. Damit wird sich auf einen Technologiestack festgelegt. Das verwendete Betriebssystem ist z/OS, als Anwendungsserver steht CICS[2] und als Datenbanksystem Db2[3]. Die Messaging Lösung ist ebenfalls aus dem Hause IBM, dabei handelt es sich um MQ[4]. Um nur einige zu nennen. Unter anderem werden Programmiersprachen wie COBOL, IBM Assembler, C, aber seit einiger Zeit auch Java verwendet.

Diese Technologie hat auch eine lange Geschichte. So wurde vor mehr als fünfzig Jahren der allererste Großrechner vorgestellt. Bis in die 90iger Jahre war dieser unangefochten, jedoch verlor er zu dieser Zeit durch die verteilten Client-Server-Umgebungen an Bedeutung. Im Jahre 2020 ist allerdings die Cloud sein größter Konkurrent. Mittlerweile wird der Mainframe als ‘legacy‘- veraltet - betrachtet. Wieso also wird sich mit dieser Technologie noch beschäftigt? - Als eine Antwort: Es wird weltweit genutzt. So verarbeiten heutzutage Großrechner weltweit circa 1,2 Millionen CICS Transaktionen pro Sekunde.[5] Im Vergleich hierzu werden 63.000 Google Suchanfragen pro Sekunde abgesetzt. [6]

Die Kombination aus hohen Workload, dem speziellen Technologiestack und dem Ruf eines veralteten Systems entspringen Risiken. Es wird immer schwieriger Nachwuchs in diesem Bereich zu finden. Zum einem da es kaum noch an Universitäten gelehrt wird. Die Seite des Hochschulkomasses[7] liefert sogar weder für ‘Mainframe‘ noch für ‘Großrechner‘ keine

[1] [**?**]
[2] Beschreibung im Absatz **??** zu finden
[3] eine relationale Datenbank
[4] Beschreibung im Absatz **??** zu finden
[5] [**?**]
[6] [**?**]
[7] [**?**]

Treffer. Zum anderen werden die momentanen Wissensträger älter und scheiden nach und nach aus. Außerdem holt sich eine Firma, die sich einen IBM Großrechner mit z/OS anschafft oder diesen betreibt, den oben genannten speziellen Technologiestack automatisch mit in die Firma. Mangels guter alternativen folgt eine starke Herstellerabhänigkeit.

Offentsichlich betreiben dennoch einige Firmen einen IBM Großrechner. Darunter zählen unter anderem Banken, das Gesundheitswesen, Versicherungen, Fluggesellschaften usw. Der gemeinsame Nenner dieser Unternehmen ist eine Massendatenverarbeitung, ein hoher Sicherheitsstandard, die dazugehörige Ressourcenverwaltung und Breitband Kommunikation. All diese Punkte sprechen für die Nutzung eines Großrechners, auch bei der DATEV eG. [**?**]

Die DATEV eG wurde am 14.02.1966 von 65 Steuerbevollmächtigten gegründet. Sie verfolgten mit der Gründung das Ziel Buchführungsaufgaben mit Hilfe der EDV zu bewältigen. Aufgrund hohen Mitgliederwachstums wurde hierfür 1969 in einen firmeneigenen IBM-Großrechner investiert.[**?**] Heute umfasst das Leistungsspektrum der DATEV eG unter anderen das Rechnungswesen, Personalwirtschaft, Consulting, IT-Sicherheit, Weiterbildung. Ein nicht unbeträchtlicher Teil, der betriebswirtschaftlichen Anwendungen laufen bis heute auf einem IBM Großrechner. So werden pro Tag circa 150.000 Batch Jobs[8] und circa 90 Millionen CICS-Transaktionen verarbeitet. Diese Last wird von circa 14.000 aktiven Modulen erzeugt. Wie in der Abbildung **??** zu sehen ist, ist COBOL mit circa 46% Prozent die am häufigsten verwendete Programmiersprache am Großrechner bei der DATEV eG. Durch diese Module werden unteranderem im Monat circa 13 Millionen Lohnabrechnungen erstellt und circa 1 Millionen Umsatzsteuer-Voranmeldungen durchgeführt.

Ich persönlich habe eine Ausbildung zum Fachinformatiker der Anwendungsentwicklung im Großrechnerbereich bei der DATEV eG abgeschlossen und parallel dazu ein Studium der Informatik begonnen. Während dieser Ausbildung lernte ich unter anderem die Entwicklungsprozesse in diesem Bereich kennen. Kurz bevor ich die Ausbildung begonnen hatte, wurde eine auf eclipse basierende Entwicklungsumgebung für COBOL und IBM Assembler in der DATEV eG eingeführt. Zuvor wurde mit Hilfe der in Abbildung **??** gezeigten Oberfläche, die nur ein Syntaxhighlighting zur Verfügung stellt, gearbeitet. Kurz vor Abschluss meiner Ausbildung wurde git auch für COBOL und IBM Assembler Sourcen als Sourceverwaltung eingeführt. Dadurch wurde die Sourceverwaltung durch Dateistrukturen oder durch ein von DATEV eG selbstenwickeltes Tool abgelöst. Nachdem eine Änderung eingebaut wurde, war eine Absprache mit den Kollegen bezüglich der Tests unvermeidlich. Dies hatte den Grund, dass für alle Tests die gleichen Test-CICS/Db2/MQ Ressourcen zur Verfügung stehen. Falls zum Beispiel eine neue Db2 Tabelle oder Datenbank benötigt wurde, musste ein manueller Prozess mit Kommunikation zwischen mehreren Abteilungen angestoßen werden. Dieser ist dadruch fehleranfällig und langsam.

[8] Beschreibung in Absatz **??**

Nach dem Abschluss meiner Ausbildung bin ich noch immer als Werkstudent bei der DATEV eG tätigt. Zwischen zeitlich habe ich auch bei einem ‘Cloud Native‘ Projekt mitarbeiten dürfen. Hier wurden mir die Vorteile dieser Umgebung bewusst. So hatte jeder Entwickler seine eigene Testumgebung und konnte ungestört von seinen Kollegen, ein neues Feature testen. Die Datenbank konnte per Knopfdruck ersetzt werden, ganze Laufzeitumgebungen konnten so erzeugt werden. Hier kam die Frage auf, ob es möglich wäre diese Prozesse auch für legacy z/OS Anwendungen einsetzen zu können. In Zusammenarbeit mit der Bereich der Technologiestrategie ergab sich das Thema dieser Arbeit.

Ist es generell technisch möglich mit dem ‘IBM Cloud Provisioning and Management for z/OS‘-Toolkit Laufzeitumgebungen automatisiert bereitzustellen? Wird dadurch der aktuelle Bereitstellungsprozess schneller und auch sicherer? Erzeugt die Nutzung einen Mehrwert bei den Stakeholdern, also den Entwicklerteams und den Administratorenteams.

Um diese Fragen zu beantworten wird die Provisionierung einer Laufzeitumgebung anhand einer speziellen Anwendung untersucht. Die Anwendung sollte ein CICS als Anwendungsserver, eine Db2 Datenbank und MQ als Messaginglösung benötigen. Um für alle drei Breiche eine Aussage treffen zu können. Die genaue Vorgehensweise wird im Kapitel **??** beschrieben.

Kapitel 2

Grundlagen

In diesem Kapitel werden für diese Arbeit wichtige Begriffe erläutert.

2.1 Mainframe / Großrechner

Mainframe und Großrechner werden in dieser Arbeit gleichbedeutend verwendet. Im modernen Sprachgebrauch kann ein Großrechner als größte zur Verfügung stehende Serverart betrachtet werden. Er wird von Unternehmen verwendet, um kommerzielle Datenbanken, Transaktionsserver und Anwendungen, die einen hohen Grad an Sicherheit und Verfügbarkeit benötigten, zu hosten. Im Gegensatz zu verteilten Serversystemen, bei denen die Funktionalitäten auf einzelne Server, wie zum Beispiel einen E-Mail-Server, einen Datenbank-Server, einen Web-Server usw. aufgeteilt sind, handelt es sich bei einem Mainframe um ein zentralisiertes System. [**?**]

2.1.1 Batch

Batch beziehungsweise Batch-Verarbeitung bezeichnet in der IT die sog. „Stapelverarbeitung“. Das heißt, dass Programme mit minimalem menschlichen Eingreifen nacheinander abgearbeitet werden. Dies geschieht meist zu einer vorher festgelegten Zeit, gesteuert wird dies über sog. „Scheduling“-Systeme. Zum Beispiel wird einmal am Tag zu einer ganz bestimmten Uhrzeit die tägliche Bewertung[1] der DATEV Rechnungsschreibung durchgeführt. Die auszuführenden Programme laufen in sogenannten „Batch-Jobs“. [**?**]

2.1.2 Batch-Job / Job

In einem Batch-Job, in dieser Arbeit wird „Job“ gleichbedeutend verwendet, wird dem System mitgeteilt welches Programm mit welchen Ein- und Ausgabedateien und Parametern

[1] Beschreibung in Absatz **??**

gestartet werden soll. DIe Skriptsprache, die diese Jobs definiert, ist im IBM-Mainframe-Umfeld die sog. „Job Control Language“, kurz JCL. Die drei Grundbausteine der JCL werden im Folgendem beschrieben.

Zunächst ist ‘JOB‘ zu nennen, auch Jobkarte genannt. Hier wird der Name des Jobs, Berechnungsinformationen, maximal zur Verfügung stehende CPU-Zeit und weitere Job-weite Parameter gesetzt.

Innerhalb eines Jobs wird mit Hilfe des „EXEC“ Befehls dem System mitgeteilt, welches Programm gestartet werden soll. Es können mehrere „EXEC"Befehle in einem Job vorkommen, dabei wird jeder einzelne als sogenannter „Job step“ bezeichnet. Dabei können dem Programm neben den Ein-/Ausgabedateien auch weitere Parameter übergeben werden.

Als letztes ist der „DD“ Baustein zu nennen. „DD“ steht für Data Definition. Ein DD-Statement verknüpft den sog. DD-Namen mit einer Datei oder einem I/O Gerät und ist somit ein Alias für diese. Ein „DD“ Baustein ist immer an ein „EXEC“ Befehl gebunden. Einem „EXEC“ können mehrere „DD“ Bausteine zugeordnet sein. [**?**]

2.2 Customer Information Control System

Das Customer Information Control System, kurz CICS, ist ein Applikationsserver für einen IBM-Großrechner mit Betriebssystem z/OS und damit eine IBM Middleware. Ein Applikationsserver stellt eine Umgebung zur Verfügung, in der Anwendungen gehostet werden können. Dabei kümmert sich dieser unter anderem um Transaktionalität, Webkommunikation und Sicherheit. Hierfür stellen Applikationsserver eine API zur Verfügung. CICS hat einen Vorteil gegenüber anderen Anwendungsservern, es unterstützt verschiedene Programmiersprachen. CICS ist ein Multi-Language Application Server und unterstützt z.B. COBOL, Assembler, Java und PLI. So können Programme innerhalb einer Anwendung in der für ihren Use-Case am besten geeigneten Sprache implementiert werden. [**?**]

2.2.1 CICS Transaktion

Ein Businessablauf wird im CICS in einer Transaktion gekapselt. Eine Transaktion kann mehrere Programme unterschiedlicher Programmiersprachen umfassen und wird über eine eindeutige „TransaktionsID“ identifiziert..

Über die TransaktionsID wird der Ablauf gestartet. Dies kann sowohl per Webanfrage oder per Messaging Queue als auch aus einem anderen Programm heraus oder manuell geschehen. In der Transaktion werden alle Änderungen, die Programme an Resourcen, wie zum Beispiel einer Datenbank oder Dateien tätigen, protokolliert. So wird im Fehlerfall die Möglichkeit eines Rollbacks sichergestellt. [**?**]

2.2.2 Voraussetzungen

Bei der DATEV eG kümmert sich ein Team, die „CICS Administration“um das Erstellen einer CICS-Instanz, starten dieser Instanz und ist generell für alles rund um die Administration des CICS Transactions Servers zuständig. Der Fokus dieser Arbeit liegt auf dem Erstellen („Provisionieren“) einer CICS-Instanz. Es werden nur die dafür notwendigen Voraussetzungen dargelegt. Außerdem liegt der Fokus auf Entwicklungs-CICS-Systemen, auf der sog. Entwicklungs-Stage, nicht auf den produktiven CICS-Systemen der DATEV eG. Aus diesem Grund werden nur Schritte, die für ein solches Testsystem benötigt werden, dargestellt. Eine weitere Rahmenbedingung besteht darin, dass nur die Arbeitsschritte, die mit z/OSMF[2] automatisiert werden, erläutert werden.

2.2.3 Einrichtung CICS Instanz

Die in diesem Absatz benötigten Informationen stammen aus Gesprächen mit Mitarbeiter 2 aus der Abteilung, die für die CICS Administration zuständig ist. Um eine lauffähige CICS Instanz den Vorausetzungen aus dem Absatz **??** entsprechend einzurichten, sind mehrere Schritte notwendig. Diese werden im Folgendem beschrieben.

2.2.3.1 CICS spezifische Dateien

Zunächst müssen CICS spezifische Dateien im z/OS angelegt werden. Im Fall des dieser Arbeit zugrunde liegende Beispiels handelt es sich um 17 verschiedene Dateien. (Kommentar VSAM? QSAM?) Diese Dateien benötigt die CICS Instanz um zum Beispiel Systemfehler zu protokollieren oder den Debugger aktivieren zu können.

2.2.3.2 CSD

In der Datei „CICS System Defintion“, kurz CSD, muss jede Ressource, die dem System zur Verfügung stehen soll, definiert werden. Eine CSD Datei kann für mehrere CICS Instanzen verwendet werden. Eine CSD Datei besteht aus mehreren Einträgen. Ein Eintrag besteht aus einer Gruppe und einer Liste. Die Gruppe ist hierbei die Definition einer Systemressource und muss manuell angelegt werden. Bei der Liste handelt es sich um das System, welches diese Ressource benötigt. Dort ist unter anderem für jede CICS Instanz hinterlegt, zu welchem Db2 Datenbanksystem und welchem MQ Messagingsystem sich diese Instanz verbinden soll.

[2] Beschreibung in Absatz **??**

2.2.3.3 STC Job

Bei einem Started Task Control-Job, kurz STC Job, handelt es sich um einen Batch Job, der mit Hilfe des „START“-Konsolenkommandos innerhalb von z/OS gestartet werden kann. Dieser Batch Job wird deshalb auch als Started Task bezeichnet.[**?**] Bei der DATEV eG existiert für jedes CICS ein solcher Job. In diesem werden zunächst einige zur Laufzeit benötigten Bibliotheken und Dateien eingebunden, unter anderem die CICS spezifischen Dateien[3]. Außerdem werden hier die SIT [4] Parameter definiert. Zunächst wird festgelegt welche Standard SIT verwendet werden soll. Anschließend können diese Standardwerte überschrieben werden. Zu diesen Parameter zählen unter anderem der eindeutige Name der CICS Instanz, der Speicherort der dazugehören CSD und die Information, ob eine Verbindung zu einem Db2 Datenbanksystem hergestellt werden soll.

2.2.4 Entfernung CICS Instanz

Um eine CICS Instanz zu entfernen muss diese zunächst gestoppt werden. Dies ist über das „STOP“-Konsolenkommando von z/OS möglich. Anschließend müssen alle im Absatz **??** beschriebene Schritte rückgängig gemacht werden. Also müssen die für diese Instanz spezifischen Dateien, die Einträge für die CICS Instanz aus der CSD Datei und schließlich auch der STC Job gelöscht werden.

2.3 IBM MQ

IBM MQ ist eine Messaging-Lösung der IBM. Diese ermöglicht den asynchronen Datenaustausch zwischen Anwendungen mittels sogenannter Queues. Alle IBM MQ Begrifflichkeiten, die in dieser Arbeit verwendet werden, werden im Folgenden erläutert. [**?**]

2.3.1 Queue Manager

Bei einem Queue Manager handelt es sich um die zentrale Ressource eines IBM MQ Systems. Er verwaltet alle anderen IBM MQ Ressourcen. Dazu gehören unter anderem die Speichersteuerung der Daten und die Wiederherstellung dieser im Falle eines Fehlers. Desweiteren koordiniert er den Zugriff aller Anwendungen auf die Nachrichten in den von ihm verwalteten Queues. Um hierbei die Konsistenz sicherzustellen, sorgt er für Locking und die notwendige Isolation der Queues. [**?**]

[3] Beschreibung in Absatz **??**
[4] CICS system initialization table

2.3.2 Queues

In Queues werden die Nachrichten, die von Programmen gesendet und gelesen werden gespeichert. Es gibt verschiedene Arten von Queues, die im Kontext dieser Arbeit relevanten Queues sind folgende:

Die Local Queue. Dabei handelt es sich um die einzige Queue Art, bei der die Nachrichten physikalisch gespeichert werden. Die anderen Queue Arten nutzen als Basis immer eine Local Queue.

Die sogenannte „Initiation Queue“, eine spezielle Art von Local Queues. Diese dient dem Queue Manager dazu, um unter bestimmten Bedingungen eine Trigger-Nachricht darauf zu schreiben. So kann eine andere Local Queue so definiert sein, dass sobald eine Nachricht auf sie geschrieben wird eine solche Trigger-Nachricht erzeugt werden. Dies ermöglicht, dass Anwendungen nur starten, wenn wirklich Daten zum Verarbeiten vorhanden sind. [**?**]

2.3.3 Process

Für das Auslösen von Anwendungen wird nicht nur die Initiation Queue benötigt, sondern auch sogenannte „Processes“. So muss der Local Queue, die den Start einer Anwendung auslösen soll, bei der Definition nicht nur die Initiation Queue bekannt gemacht werden, sondern auch ein Process. Ein Process legt den „Type“ und den Namen der zu startenden Anwendung fest. Als „Type“ können beispielhaft CICS oder auch WINDOWSNT für Windows unterstütze Platformen genannt werden. Ist der „Type“ CICS, muss der Name der Transaktion angegeben werden, für Windows Platformen der Dateipfad der auszuführenden exe. [**?**]

2.4 Stages, Sysplexe und Logical Partitions

Innerhalb der DATEV eG existieren vier sogenannte „Stages“ für die Entwicklung auf dem Mainframe. Dabei handelt es sich um abgekapselte Systemumgebungen. Eine Stage besteht aus einer oder mehreren „Logical Partitions “, kurz LPARs, mit eigenen Subsystemen und eigener Ressourcenverwaltung. Zu den Subsystemen zählen unter anderem CICS [5], Db2[6] und IBM MQ[7]. Dadurch wird eine strickte Trennung zwischen den Stages realisiert. Die vier Stages werden im Folgenden beschrieben.
„Entwicklung“

[5] Beschreibung in Absatz **??**
[6] ein relationales Datenbanksystem für z/OS
[7] Beschreibung in Absatz **??**

Stage auf der alle z/OS Entwickler an neuen Features arbeiten und erste kleinere Tests durchführen.

„Qualitätssicherung“
Hier werden vor allem Integrationstests mit extra dafür erstellen Testdaten durchgeführt.

„Vorproduktion“
Die Vorproduktion steht für weitere Integrationstests zur Verfügung. Jedoch produktionsnäher und auch mit Echtdaten, also Daten aus der Produktion.

„Produktion“
Hier befindet sich die Software, die von den Kunden verwendet wird. Ein Programm muss die Entwicklungs-, Qualitätssicherungs- und Vorproduktionsstages durchlaufen bevor es dem Kunden in der Produktion zur Verfügung gestellt wird.

Die LPARs der vier Stages sind auf zwei sogenannte „Sysplexe“ aufgeteilt. Ein Sysplex kümmert sich um die Kommunikation zwischen den LPARs der einzelnen Stages und verwaltet Ressourcen LPAR übergreifend. [**?**]

Neben den oben erwähnten zwei Sysplexen existiert noch ein weiterer Sysplex. Diese werden alle im Folgenden beschrieben.

„LAB/Test-Plex“
Der Testplex ist als Labor für Änderungen am System zu betrachten. So werden zum Beispiel neue Betriebssystemsversionen auf diesem geprüft.

„QS-Plex“
Enthält unter anderem die LPARs der Entwicklung und der Qualitätssicherung.

„Prod-Plex“
Im Prod-Plex sind die LPARs der Vorproduktion und der Produktion enthalten.

2.5 IBM Cloud Provisioning and Management for z/OS

In diesem Absatz wird zunächst auf die für dieses Kapitel grundlegenden Begriffe eingegangen. Anschließend wird das Produkt „IBM Cloud Provisioning and Management for z/OS“ erläutert.

2.5.1 Begrifferklärung

Im Folgenden werden einige allgemeine Begriffe, die im Umfeld des Tools „IBM Cloud Provisioning and Management for z/OS“ vorkommen, erläutert.

2.5.1.1 Provisioning

Ins Deutsche übersetzt bedeutet es Bereitstellung, in dieser Arbeit wird auch Provisionierung verwendet. In dieser Arbeit umfasst dieser Begriff die Bereitstellung einer Laufzeitumgebung, beziehungsweise den Prozess, der hierfür benötigt wird.

2.5.1.2 Workflow

Ein Workflow ist eine beliebig komplexe, eindeutige Aneinanderreihung von sogenannten Steps. Nach der Ausführung dieser Steps wird ein bestimmtes Ziel erreicht, zum Beispiel die erfolgreiche Bereitstellung eines CICS Systems. Für die Definition eines Workflows, der dazugehörigen Steps und ihrer Variablen wird XML genutzt. Ein Step beschreibt einen Teilablauf eines Workflows. Innerhalb eines Steps können sowohl interne und externe Scripte als auch JCLs und somit Programme ausgeführt werden. Darüber hinaus besteht die Möglichkeit REST-Calls auszuführen. Es können Bedingungen für die Durchführung eines Steps definiert werden. So ist es zum Beispiel möglich, einen Step nur durchzuführen, wenn eine bestimmte Variable einen bestimmten Wert besitzt. Es wird ein XML Schema verwendet um sicherzustellen, dass in dem XML-Skript zur Laufzeit keine syntaktischen Fehler vorhanden sind. [**?**]

Ein Nachteil von Workflows ist, dass diese statisch sind, das heißt, dass die Variablenzuweisungen immer zum Zeitpunkt der Erstellung stattfindet. Dadurch ergibt sich, dass für jede kleine Änderung ein eigener Workflow erzeugt werden muss. Somit ist ein Workflow eher ein Einmal- bzw. Wegwerfprodukt.

2.5.1.3 Template

Bei dem Nachteil von Workflows als Wegwerfprodukt setzen die sogenannten Templates an. Ein Template besteht aus drei Dateien.

1) Datei für Eingabevariablen. In dieser Datei kann Workflowvariablen Werte zugewiesen werden. Diese Variablen müssen bei ihrer Definition im Workflow entsprechend gekennzeichnet sein.

2) Die sogenannte Aktion-Definitions-Datei. Hier werden die Aktionen, die ein Anwender mit diesem Template durchführen kann, festgelegt. Einer Aktion wird eine Workflow Definitions Datei und somit ein Workflow zugewiesen. Darin ist die Datei für die Eingabevariablen und die Festlegung, welche Variablen davon verwendet werden, anzugeben.

3) Die Manifest-Datei. In dieser wird dem Template mitgeteilt, an welchem Speicherort sich die oben genannten Dateien befinden. Da ein Template immer provisioniert werden kann,

wird hier auch der Speicherort des Bereitstellungsworkflows angeben. Zusätzlich kann noch eine Beschreibung des Templates hinzugefügt werden.

Somit bildet ein Template einen Rahmen um mehrere Workflows und ermöglich eine schnellere De-/provisionierung. Das hat den Vorteil, dass Variablen nur an einer Stelle geändert werden. Daüber hinaus besteht die Möglichkeit, per Awendereingabe den Variablen zum Zeitpunkt der Provisionierung einen Wert zuzuweisen. Somit ist ein Template deutlich flexibler als ein Workflow. [**?**]

2.5.1.4 Instance

Hierbei handelt es ich um das Ergebnis nach der Provisionierung eines Templates, zum Beispiel um ein funktionsfähiges CICS.

2.5.2 IBM Cloud Provisioning and Management for z/OS

Das IBM Cloud Provisioning and Management for z/OS bietet die Möglichkeit, mehrere Systeme (sog. Middleware) innerhalb eines z/OS Betriebssystems zu provisionieren, unter anderem Laufzeitumgebungen wie CICS. Für diese Aufgaben stehen zwei Schnittstellen zur Verfügung. Zum einen das z/OS Provisioning Toolkit, (z/OSPT) und zum anderen z/OS Management Facility (z/OSMF). [**?**]

2.5.2.1 z/OS Provisioning Toolkit

z/OSPT bietet ein Kommandozeileninterface für die Bereitstellung und das Verwalten von Laufzeitumgebungen. In Abbildung **??** werden die möglichen Kommandozeilenbefehle mittels des Befehls „zospt -h“ in einem Kommandofenster angezeigt. Mit z/OSPT werden noch zwei weitere Begriffe eingeführt.
„Images“. Dabei handelt es sich grundsätzlich um ein Template, jedoch kann dieses Template über eine weitere Inputdatei verändert werden. Dadurch kann ein Template mit spezifischen Änderungen provisioniert werden, ohne dass ein neues Template erzeugt werden muss. Dies erhöht die Flexibilität der Templates weiter.

„Container“. Dabei handelt es sich eins zu eins um eine „Instance“[8]. [**?**]

[8] Beschreibung in Absatz **??**

```
$ zospt -h
IBM z/OS Provisioning Toolkit V1.1.5

Usage: zospt [OPTIONS] COMMAND [arg...]

Options:
 --version   : Displays the command line version.
 -h (--help) : Displays the command line help.

Commands:
    build       PATH [-h (--help)] -t (--tag) <imageName>              Build an image
    images      [-h (--help)]                                          List all images
    inspect     <imageName> | <containerName> | <containerId>          Inspect an image or a container
                [-h (--help)]
    rm          <containerName> | <containerId> ... [-f (--force)]     Remove one or more containers
                [-h (--help)]
    rmi         <imageName> ... [-h (--help)]                          Remove one or more images
    run         <imageName> [--draft]                                  Run an image in a new container
                [--link <containerName> | <containerId>:<alias>]
                [--name <containerName>] [-h (--help)] [-q (--quiet)]
    start       <containerName> | <containerId> ... [-h (--help)]      Start one or more containers
    stop        <containerName> | <containerId> ... [-h (--help)]      Stop one or more containers
    ps          [-a (--all)] [-f (--filter) <filter>] [-h (--help)]    List containers

Run 'zospt COMMAND --help' for more information on a command.
```

Abbildung 2.1: z/OSPT mögliche Kommandozeilenbefehle

2.5.2.2 z/OS Management Facility

Der Hauptaugenmerk dieser Arbeit liegt bei z/OSMF, da damit die Verwaltung von Workflows und Templates über eine browserbasierende Schnittstelle möglich ist. Durch diese Oberfläche, in Abbildung **??** dargestellt, ist es einfacher zu bedienen als das Kommandozeileninterface von z/OSPT und somit wird der Einstieg in die Provisionierung erleichtert.

//hier zosmf welcomepage screenshot

Die linke Seite der Abbildung **??** zeigt den Umfang der z/OSMF Funktionen. Für diese Arbeit besitzt nur der Menüpunkt „Cloud Provisioning“ Relevanz. Unter diesem Punkt sind die Funktionalitäten für die automatisierte Bereitstellung von Templates zu finden. [**?**]

Zuerst ist das „Resource Management“ zu nennen.. Darunter werden sogenannte „Domains“ und die dazugehörigen „Tenants“ verwaltet. Unter einer „Domain“ ist ein System zu verstehen, das Systemressourcen in Ressourcenpools gliedert. „Tenants“ sind die dazugehörigen Rechtegruppen, die dem Anwender den Zugriff auf und die Nutzung von zugeordneten Templates ermöglicht. Einem Template muss sowohl eine„Domain“ als auch ein „Tenant“ zugewiesen werden. [**?**]

Zur Verwaltung der Templates und Instances kommen die „Software Services“ zum Einsatz. Dort können neue Templates über die „Manifest Datei“ hinzugefügt werden. Dann muss, wie oben beschrieben, eine „Domain"und ein „Tenant“ zugwiesen werden. Anschließend kann das Template, falls es keine Fehler beinhaltet, veröffentlicht werden. Es ist zu empfehlen vorher einen „Test Run“ durchzuführen. Dabei wird eine Instance testweise provisioniert. Diese Instance verhält sich genauso wie eine Instance, die aus einem veröffentlichten Template

erzeugt wurde. Somit können damit das Template und die in der Aktion-Definitions-Datei definierten Aktionen getestet werden. [**?**]

Kapitel 3

Vorgehensweise

Zu Beginn dieser Arbeit wurde zunächst das ‘IBM Cloud Provisioning and Management for z/OS‘-Toolkit mit seinen Varianten zOSMF und zOSPT analysiert. Anschließend folgt die Analyse und Darstellung des Ist-Zustandes einer Beispielanwendung, inklusive der Beschreibung des momentan etablierten Bereitstellungsprozesses für CICS, Db2 und MQ. Schließlich wird an Hand einer Beispielimplementierung gezeigt, ob es technisch möglich ist eine Laufzeitumgebung automatisiert mit dem oben genannten Toolkit für diese Beispielanwendung bereitzustellen.

Im folgenden wird die Vorgehensweise während der Beispielimplementierung erläutert. So wird zunächst auf einer Systemumgebung für Tests von neuen Betirebssystemversionen oder ähnlichem, dem sogenannten Testplex, begonnen. Dieser ist komplett von anderen Systemumgebungen abgekapselt, um unvorhersehbare Fehler zu vermeiden. In dieser Umgebung wird zunächst untersucht, wie es möglich ist ein CICS zu provisionieren. Hierfür wird vorerst ein von der IBM bei der Installation von z/OSMF mitgeliefertes minimales CICS Template verwendet. Anschließend wird ein umfangreicheres mitgeliefertes Template an die Anforderungen der Anwendung angepasst. Für die Erstellung von Db2 Datenbanken existiert innerhalb der bereits eine REST-API. Es wird versucht diese innerhalb des Templates zu nutzen. MQ Queues werden mittels Standard IBM Jobs provisioniert. Da im Testplex jedoch keine Anwendungsdaten vorhanden sind, wird vorerst nur die benötigten Systeme provisioniert, um so die Grundvoraussetzungen zu schaffen.

Nachdem eine CICS Instanz, eine Db2 Datenbank und MQ Queues auf dem Testplex sowohl provisioniert als auch deprovisioniert werden können, folgt der nächste Schritt. Dabei handelt es sich um den Wechsel der Systemumgebung vom Testplex auf die Entwicklungssystemumgebung. Letzteres ist die Testumgebung für alle Mainframe Entwickler. Hier werden vor allem neue Programmversionen entwickelt und damit kleinere Tests durchgeführt. Außerdem sind in der Entwicklungssystemumgebung alle Anwendungsdaten, die für diese Tests notwendig sind, vorhanden. Somit kann die Integrierung der Beispielanwendung in das provisionierte CICS stattfinden.

Zuletzt werden Interviews mit Vertretern aus dem Administratorenteam, Entwicklerteam und dem Team für die Technologiestrategie, abgehalten. Dadurch erfolgt eine Bewertung

bezüglich des möglichen Nutzwertes und ob die Technologie von den Stakeholdern akzeptiert wird.

Kapitel 4

Analyse

Im Folgendem erfolgt eine Beschreibung der Beispielanwendung ‘Rechnungsschreibung‘. Die dafür benötigten Informationen stammen aus Gesprächen mit Mitarbeiter 1 aus der Abteilung, die für die Rechnungsschreibung zuständig ist. Hierbei wird vor allem der technische Aspekt beleuchtet. Anschließend wird der aktuelle Bereitstellungsprozess für Laufzeitumgebungen, den dazugehörigen Datenbanksystem und einer Messaging Lösung dargestellt.

4.1 Rechnungsschreibung

Für diese Arbeit wurde die Rechnungsschreibung als Beispielanwendung herangezogen, weil sie folgenden Anforderungen entspricht. Es handelt sich zum einem um eine in sich abgeschlossene Anwendung, die nur zu Beginn des Prozesses von anderen Anwendungen abhängig ist. Zum anderen benötigt die Rechnungsschreibung ein CICS als Laufzeitumgebung, eine Db2-Datenbank und MQ als Messaginglösung. Somit kann ein umfangreicher Bereitstellungsmechanismus in dieser Arbeit untersucht werden.

4.1.1 Beschreibung

Die Erzeugung der Rechnungen lässt sich in mehrere Schritte unterteilen, gesammelt werden diese Schritte als Rechnungsschreibung bezeichnet.
Bei dem Ablauf handelt es sich um einen Batch[1]-Ablauf, der auf einem Großrechner läuft. Nur die Preisermittlung wird in ein CICS ausgelagert. Zunächst wird nach jeder kostenpflichtigen Leistungserbringung durch die dazugehörige Anwendung ein Berechnungssatz erzeugt. Ein Berechnungssatz beinhaltet die Metainformationen der Berechnung unter anderem die Artikelnummer, Menge und den Ordnungsbegriff. Der Preis und der Rechnungsempfänger wird zu einem späteren Zeitpunkt innerhalb der Rechnungsschreibung ermittelt.

[1] Stapelverarbeitung

4.1.1.1 Einpflegung Berechnungssätze

Für das Einpflegen der Berechnungssätze in den Rechnungsschreibungsablauf stehen den Anwendungen drei Möglichkeiten zur Verfügung.
Bei der Ersten Möglichkeit handelt es sich um die Verwendung des DMVINF[2]-Moduls und der dazugehörigen Schnittstelle. Dieses Modul ist in der Programmiersprache Assembler entwickelt worden. Das Ergebnis dieses Moduls ist eine sequenzielle Datei am Großrechner, dieses Format lässt sich mit einer .txt Datei unter Windows vergleichen. Diese Datei, auch Berechnungsdatei genannt, hat folgenden Aufbau. Der erste Satz enthält Steuerinformationen, wie zum Beispiel Datum/Uhrzeit, Produkt usw. Danach kommen die eigentlichen Berechnungssätze. Schließlich folgt noch die Anzahl der Sätze und die Summe der einzelnen Artikel in einem Satz mit Kontrollinformationen. Diese Kontrollinformationen werden im weiteren Verlauf mit den eingelesenen Werten abgeglichen, dadurch wird Datenverlust und unkontrollierte Eingriffsmöglichkeiten von außen ausgeschlossen. Aus dem Aufbau einer solchen Datei lässt schließen, dass verschiedene Schritte für die Erzeugung innerhalb der Anwendung notwendig sind.

Für die nachfolgenden Schritte stellt das DMVINF-Modul jeweils Schnittstellen zur Verfügung. Zuerst wird beim sogenannten Open die Datei erstellt und der Steuersatz geschrieben. Danach folgt das eigentliche Schreiben der Berechnungssätze, dabei dürfen nur bestimmte Felder (Ordnungsbegriffe, Länderschlüssel und Mengen) verändert werden. Um unzulässige Veränderungen zu verhindert, haben diese einen Abbruch der Verarbeitung zur Folge. Schließlich folgt noch der ‘Close‘ bei dem die Kontrollinformationen geschrieben werden. Hinzuzufügen ist, dass die variablen Informationen einer Formalprüfung unterzogen werden. So entstehen je nach fachlicher Logik und Laufhäufigkeit der Anwendung mehrere Berechnungsdateien.

Eine weitere Möglichkeit die Berechnungsinformationen in den Ablauf einzupflegen ist die Übergabe über einen mit der Programmiersprache Java realisierte WebService. Hier werden die Berechnungsinformationen im XML-Format bereitgestellt. Das Ergebnis der entsprechenden Plausibilitätsprüfungen, die in einem Onlineverfahren durchgeführt werden, wird direkt an die aufrufende Anwendung zurückgegeben. Sind die Daten korrekt werden diese vorerst in einer Datenbank gespeichert. Vor dem nächsten Schritt wird diese Datenbank ausgelesen und mit der ersten Möglichkeit in den Kernablauf eingespeist.

Bei der letzten Möglichkeit handelt es sich um die Übergabe mittels einer CSV-Datei. Die Datei wird auf den Großrechner übertragen und dort mit dem DMVINF-Modul verarbeitet. Dieses Verfahren wird kaum von produktiven Anwendungen sondern hauptsächlich für Test- oder Qualitätssicherungszwecke genutzt.

[2] DatevMakroVerarbeitungsinformation

Mittels dieser drei Möglichkeiten werden insgesamt monatlich circa 30 Millionen Datensätze bereitgestellt und weiterverarbeitet. Diese Datensätze stehen innerhalb der durch das DMVINF-Modul erzeugten Berechnungsdateien dem weiteren Verlauf als Input zur Verfügung. Um sicher zu stellen, dass all diese Dateien auch verarbeitet werden, wird bei Erstellung einer solchen ein Eintrag in eine Kontrolldatei vorgenommen. In dieser Kontrolldatei wird jedes Lesen und somit auch das Lesen im weiteren Verlauf gekennzeichnet. Eine monatliche Überprüfung führt die zuständige Abteilung durch.

4.1.1.2 Tägliche Bewertung

Der nächste Schritt des Rechnungsschreibungsprozesses ist die sogenannte Tägliche Bewertung. Dieser Ablauf läuft einmal täglich von Montag bis Freitag und ist für die Preis- und Rechnungsempfängerermittlung zuständig. Zur Realisierung wurden die Programmiersprachen Assembler und COBOL genutzt. Am Ende dieses Ablaufes steht die ARUBA[3]-Db2-Datenbank. Dort werden die Berechnungsdaten der letzten 36 Monate aufbewahrt. Dabei handelt es ich um insgesamt circa 3,8 Milliarden Datensätze von einer Gesamtgröße von circa 400 GB mit Indizes. Diese Datensätze beinhalten alle Informationen für die endgültige Erzeugung der Rechnungen.

Der erste Schritt der Täglichen Bewertung ist das Zusammenführen der Berechnungsdateien aus dem vorherigen Schritt und aus den bereits vorhandenen Daten des laufenden Monats aus der ARUBA-Db2-Datenbank. Zusätzlich werden während dieser Zusammenführung den Berechnungssätzen auf Basis der abgebenden Anwendung die entsprechenden Rechnungsstellungsrythmen (täglich oder monatlich) zugewiesen. Anschließend wird mit Hilfe der Beraternummer die zugehörigen Betriebsstätte-, Rechnungsempfänger-, Hauptberater- und Mitglieds- bzw. Geschäftspartnernummer ermittelt. Die Beraternummer ist als oberster Ordnungsbegriff in den Berechnungssätzen enthalten. Außerdem wird die Debitorenkontonummer entweder durch die Mitglieds- oder durch die Geschäftspartnernummer zugeordnet. Für die Preisermittlung werden die Datensätze nach Geschäftspartner gruppiert. Im DATEV eG Umfeld ist ein Geschäftspartner entweder eine Kanzlei oder ein einzelner Mandant, dieser ist jedoch meist einer Kanzlei zugeordnet.

Dann werden die gruppierten Daten auf Grund der Performance an ein CICS übertragen. Die Architektur wird in **??** beschrieben. Dort findet die Preisermittlung mit den dazugehörigen kundenindividuellen Abhängigkeiten, wie zum Beispiel Rabatte, statt. Anschließend werden die Daten wieder zurück an den Batch-Ablauf übertragen. Hier werden die Rechnungsnettobeträge geprüft, ob diese über einem bestimmten Rechnungslimitbetrag liegen. Falls dies nicht der Fall ist, werden die Berechnungssätze als BUL[4] gekennzeichnet und in die folgende Rechnungsperiode vorgetragen. Schließlich wird noch die Umsatzsteuer ermittelt.

[3] Abrechnungs- und Umsatz-Basis
[4] Berater unter Limit

Abschließend werden die neu erzeugten Berechnungsdaten in die ARUBA-Db2-Datenbank eingepflegt und entsprechend gekennzeichnet.

4.1.1.3 Rechnungsaufbereitung

Als letzter Schritt folgt die Rechnungsaufbereitung. Diese erfolgt am ersten Werktag im Monat. Mit Hilfe der ARUBA-Db2-Datenbank wird ermittelt, welchen Kunden eine Rechnung zugestellt werden muss. Außerdem wird dabei der Zustellungsweg, per Post oder E-Mail, bestimmt. Darauf folgt die Aufbereitung der Druckrohdaten und letztlich das Versenden der Rechnungen an die Berater. Zusätzlich werden die Rechnungen noch im PDF-Format archiviert.

4.1.2 Architektur der Preisermittlung

In dieser Arbeit steht das automatisierte Provisionieren von Laufzeitumgebungen im Fokus. In diesem Fall handelt es sich um die Laufzeitumgebung CICS mit den dazugehörigen Elementen. Deshalb wird im Folgenden nur darauf eingegangen.

Das System muss an Lasttagen bis zu 180.000 Geschäftspartner verarbeiten können. Um all diese an das CICS zu übertragen stehen dem System mehrere IBM MQ Queues zur Verfügung. Bevor die eigentliche Bestimmung der Preise stattfindet, werden zunächst die Listenpreise ermittelt. Hierfür werden zwei Queues verwendet. Eine startet eine Transaktion im CICS, die andere wartet auf deren Antwort. In diesem Ablauf werden die benötigten Listenpreise aus einer Datenbank ausgelesen und in einen Hauptspeicherbereich der CICS Instanz gespeichert. Dies ist auf Grund der Last auf dem System notwendig, da ein Hauptspeicherzugriff schneller als ein Datenbankzugriff ist. Somit gab es bei der Laufzeit eine Einsparung um circa XXXX Prozent.

Für die Bestimmung der Preise mit den Preisabhängigkeiten bestehen weitere Queues. Darunter ist eine allgemeine Queue in der alle Aufträge, die für die Weiterverarbeitung zur Verfügung stehen, geschrieben werden. Pro Geschäftspartner wird ein Auftrag angelegt. In diesem Auftrag befinden sich die Namen vier weiterer Queues. Eine dieser Queues beinhaltet alle Informationen, die für die Preisermittlung des dazugehörigen Geschäftspartners notwendig sind. In den restlichen drei Queues sind die Ergebnisse der Preisermittlung gespeichert. Die Queuenamen werden nicht dynamisch generiert, da dies zu Performanceproblemen führt. Deshalb existieren für jede der vier Queues jeweils 100 vorgefertigte Namen. Somit können auch maximal nur 100 Aufträge gleichzeitig auf Weiterverarbeitung warten. Falls dieses Limit erreicht ist, wartet der Batch-Ablauf solange bis einer der Aufträge fertig gestellt wird. Sobald ein Auftrag in die allgemeinen Auftragsqueue geschrieben wird, wird eine CICS-Transaktionen gestartet. Diese führt die Preisermittlung durch und schreibt das Ergebnis

auf die dazugehörigen Queues. Ist dies geschehen stehen die Queues wieder für einen neuen Auftrag zur Verfügung. Es können maximal 30 Transaktionen zeitgleich arbeiten.

Für die Preisermittlung wird auch eine Db2-Datenbank, in der die Einzelpreise der Artikel gespeichert sind, verwendet. Wenn alle Transaktionen direkt auf diese Datenbank zugreifen würden, hätte dies über 60 Millionen Datenbankzugriffe zur Folge. Dies führt zu massiven Einbußen bei der Performance. Deshalb werden bevor die eigentliche Preisermittlung stattfindet, alle benötigten Einzelpreise und Preisabhängigkeiten ermittelt. Diese Informationen werden dann in einen sogenannten 'SHARED GETMAIN'-Bereich gespeichert. Dabei handelt es sich im Prinzip um einen Hauptspeicherbereich des Großrechners. Die Adresse von diesem Bereich wird dem Ablauf zur Verfügung gestellt. Somit greifen die einzelnen Transaktionen nicht mehr direkt auf die Datenbank zu, sondern auf den schnelleren Hauptspeicher.

4.2 Aktueller Bereitstellungsprozess

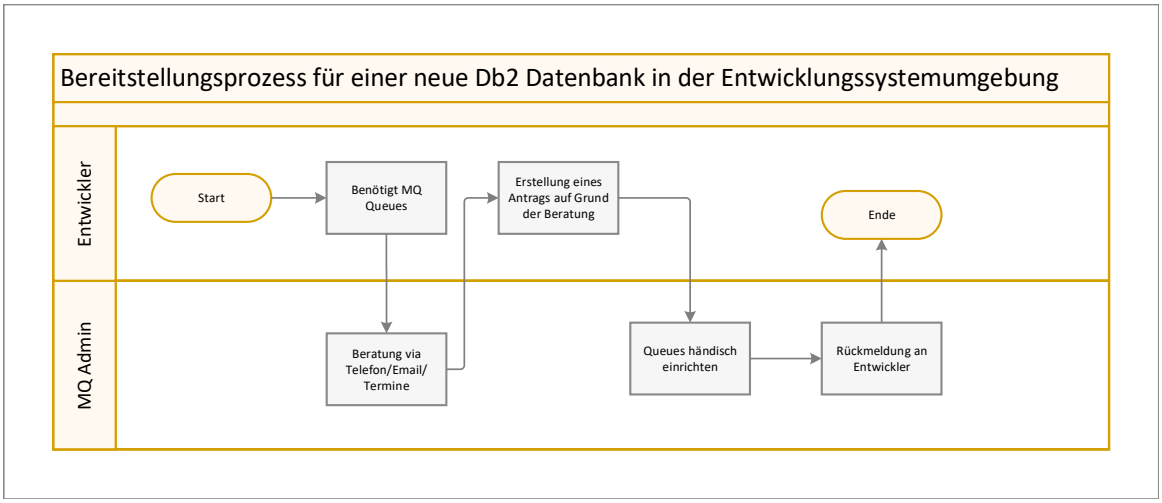

Abbildung 4.1: Bereistellungsprozess von MQ

Wie in den drei Diagrammen zu erkennen ist, ist der aktuelle Bereitstellungsprozess noch mit vielen manuellen Schritten verbunden. Außerdem ist der Hauptaufwand in den Administratorenteams angesiedelt. Das Entwicklerteam ist der Initiator des Ablaufs. So kümmert es sich um Formulare und die erste Kontaktaufnahme zum Administratorenteam. Bei dem Bereitstellungsprozess einer Db2 Datenbank muss es außerdem Projekt Informationen, unter anderem Daten der Voruntersuchung, bereitstellen. Hinzu komm das Erstellen eines Datenbankmodells. Hierfür wird Datenbankwissen benötigt.

Zusätzlich zu den vielen manuellen Schritten sind die vielen Absprachen zwischen mehreren Abteilungen zu nennen. Sobald ein beteiligtes Team nicht zu Verfügung steht, schiebt sich

der komplette Zeitplan nach hinten. Der Prozess für die Bereitstellung eines CICS Systems, mit einer Db2 Datenbank und MQ Queues dauert in der Summe circa sechs Arbeitstage. Es setzt sich aus der Dauer der Einzelnenprozesse zusammen, für jedes Subsystem wird mit circa zwei Arbeitstage gerechnet. Unter den Annahmen, dass zum einen jedes beteiligte Team nur diese Aufgabe zu erledigen hat und zum anderen jeweils ein Tag für die Beratung durch die Administratorenteams beanschlagt wird. Natürlich ist ein parallelisierter Ablauf der einzelnen Teilprozesse möglich, so kann die Gesamtdauer im besten Fall auf circa zwei bis drei Arbeitstage verkürzt werden.

Ein weiterer Punkt ist, dass die Kommunikation beziehungsweise der Initiator für den Start des gesamten Prozesses meist per Zuruf stattfindet. So existiert für die erste Kontaktaufnahme kein Formular, keine Automatition oder ähnliches. Zur Kommunikation wird auf E-Mail, Telefon oder mittels Terminen zurückgegriffen.

Kapitel 5

Realisierung

In diesem Kapitel wird beschrieben, wie die Aufgabe dieser Arbeit gelöst wurde. Dazu wird nach der im Kapitel **??** beschriebenen Reihenfolge, der Arbeitsschritte vorgegangen. Es ist nochmal zu erwähnen, dass zunächst die Provisionierung einer CICS Instanz untersucht wird. Danach wird in weiteren Schritten zuerst eine Db2 Datenbank und schließlich MQ Queues dem Bereitstellungsprozess hinzugefügt. Die Funktionsfähigkeit der so generierten Laufzeitumgebung wird mittels eines Testablaufes sichergestellt. Zuletzt wird noch ein Fazit zur Realisierung gezogen und ein Ausblick im Bezug auf die technischen Aspekte gegeben.

5.1 Testplex

Vor Beginn der eigentlichen Untersuchung mussten zunächst alle benötigten Rechte beantragt werden. Hierzu zählen unter anderem die Rechte für die Nutzung des Testplexes, die Nutzung von z/OSMF und z/OSPT und die Rechte für die Templateverwaltung innerhalb von z/OSMF. Außerdem war es auf dem Testplex möglich, die Rechte für das Erstellen der CICS Dateien, das Recht, um ein CICS starten zu dürfen und die Rechte für die Administration von Db2 und MQ einer persönlichen UserID zu geben. Dies stellt kein Problem dar, weil es sich bei dem Testplex um eine reine Systemtestumgebung handelt. Außerdem benötigt das IBM Cloud and Management for z/OS lesenden Zugriff auf den Speicherpfad der Template Dateien. Schließlich konnte mit dem ersten Versuch ein bei der Installation von z/OSMF mittgeliefertes minimales CICS Template zu provisionieren begonnen werden.

5.1.1 IBM Standard CICS Template

Trotz der Vorteile durch die Weboberfläche von z/OSMF wurde zunächst auf z/OSPT gesetzt. Diese Entscheidung fiel auf Grund der höheren Flexibilität, durch Images. Da es sich um ein mitgeliefertes Template handelt, sind alle benötigten Workflow Definitionsdateien und Template Dateien vorhanden. Somit konnte der Konsolenbefehl ‘zospt build‘ auf dieses Template durchgeführt werden. Dadurch sollte ein Image erzeugt werden. Jedoch zeigte sich ein weiterer Nachteil des Kommandozeileninterfaces, es ist nicht möglich Templates

Variablenname	Kurzbeschreibung
DFH_REGION_SEC	Legt fest, ob für das CICS Sicherheit im Allgemeinen aktiviert ist.
DFH_REGION_SECPRFX	Wenn DFH_REGION_SEC gesetzt ist, legt den Namen Perfix bei Authentificationanfragen für Ressourcen fest.
DFH_REGION_APPLID	Applikations ID der zu provisionierenden CICS Instance.
DFH_LE_HLQ	High-level qualifier[1] für die Sprachumgebung[2]
DFH_REGION_HLQ	High-level qualifier für die CICS Dateien.
DFH_REGION_LOGSTREAM	Legt fest, wie die Log Dateien für das provisionierte CICS erstellt werden sollen.
DFH_STC_ID	User ID mit dem die CICS Instanz startet.
DFH_REGION_DFLTUSER	Default User ID für das CICS.
DFH_REGION_VTAMNODE	Name des VTAM Knotens, wenn das CICS hochfährt.
DFH_REGION_MEMLIMIT	Dem CICS maximal zur Verfügung stehender Speicherplatz.
DFH_ZOS_PROCLIB	Datei auf dem Großrechner, die den Job enthält, der für das Erzeugen der CICS Instanz zuständig ist.
DFH_ZOS_VSAM_VOLUME	Speichersystem auf welchem die Dateien gespeichert werden sollen. Entscheidung kann auch an das System abgeben werden.
DFH_CICS_USSHOME	Homeverzeichnes des Unix System Services
DFH_CICS_HLQ	High-level qualifier von dem CICS Installationsort.

Tabelle 5.1: Zu verändernde Variablen im minimalen CICS Template

eine Domain und einen Tenant zuzuweisen. Dies hatte zur Folge das der Befehl ‘zospt build‘ fehlschlug. Zusätzlich führte es dazu, dass für alle folgenden Aufgaben z/OSMF genutzt wird.

Das z/OSMF auf die Template- und Workflow-Dateien zugreifen kann, sind diese in einem Unix Dateisystem auf dem Großrechner gespeichert. Das Template konnte dann ohne weitere Probleme in die Software Services aufgenommen werden. Dabei wurden ihm eine Domain und ein Tenant zugewiesen. Bevor das Template provisioniert werden kann, müssen Änderungen in der Eingabevariablen Datei vorgenommen werden. Dazu mussten die Werte, der in Tabelle **??** genannten Variablen angepasst werden. Die Kurzbeschreibungen und die Beschreibungen aller Variablen, die im Standard Template vorhanden sind, ist in [**?**] zu finden. Das diese Änderungen greifen, muss das Template aktualisiert werden. Dies ist in der Oberfläche per Knopfdruck durchgeführt worden.

Als nächster Schritt wurde ein Testlauf und somit ein erster Versuch das Template zu provisionieren durchgeführt. Hierbei trat beim ersten Step, der einen Job starten wollte, ein Fehler auf. Nämlich um einen Rechte Verstoß bezüglich des Jobnames. Bei der DATEV eG benötigt eine User ID die Rechte, um Jobs mit bestimmten Namen starten zu dürfen. Da im Template von der IBM vorgeschlagene Jobnamen verwendet werden, kommt es zum Verstoß. Um dieses Problem zu lösen, wurden die Jobnamen innerhalb des gesamten Templates an DATEV eG Standards angepasst. Nachdem das Template aktualisiert wurde, wurde erneut versucht zu provisionieren. Dabei stellte sich heraus, dass der Befehl, um ein CICS zu starten innerhalb der DATEV eG einen weiteren Parameter benötigt. Dieser wurde hinzugefügt und danach funktionierte das Provisionieren und alle definierten Aktionen des minimalen IBM Standard CICS Templates.

5.1.1.1 DATEV eG spezifischen CICS Template

Nachdem das minimale IBM Standard CICS Template funktionsfähig war und erste Erfahrungen mit z/OSMF gesammelt wurden, wurde ein allgemeines mitgeliefertes Template untersucht. Wie in der Tabelle **??** dargestellt ist, ist dieses Template mit insgesamt 76 verwendeten Dateien sehr umfangreich. Zu diesen Dateien zählen alle, die direkt mit dem Template in Verbindung stehen. Das Template beinhaltet nicht nur die Möglichkeit verschiedene CICS Typen zu provisionieren, sondern auch, ob dies mit Skripten oder mit der REST-Api geschieht. Dadurch verliert das Template an Übersichtlichkeit. Zusätzlich kommt am Ende bei der Provisionierung keine CICS Instanz heraus, die einer DATEV eG spezifischen Instanz entspricht. Auf Grund dessen wurden alle für ein DATEV eG spezifischen CICS nicht notwendigen Dateien entfernt. Wie in der Tabelle **??** zu sehen ist, hatte das unter anderem die Löschung von knapp der Hälfte der Dateien zur Folge. Des Weiteren wurden auch viele nicht benötigte Variablen und Steps entfernt. Dadurch schrumpft die provision.xml um circa ein Drittel. Zusammenfassend lässt sich sagen, dass das Template an Übersichtlichkeit gewonnen hat.

Als nächstes wurde mit der Modifizierung der restlichen Dateien begonnen. Als Ziel davon steht eine funktionsfähige DATEV eG spezifische CICS Instanz. Zunächst wurden die Namen der CICS Dateien[3] an die DATEV eG internen Namenskonventionen angepasst.

In Zusammenarbeit mit den CICS Administratorenteam wurde festgelegt, dass eine jede provisionierte CICS Instanz ihre eigene CSD Datei zur Verfügung gestellt bekommen soll. Hierfür soll die bestehende von den Kollegen gepflegte Datei kopiert und mit bestimmten Namenkonventionen gespeichert werden. Somit ist sichergestellt, dass durch die neu provisionierten Instanzen die Alten nicht beeinflusst werden. Außerdem kann jeder Anwender

[3] Beschreibung in Absatz **??**

so ohne Seiteneffekte seine CSD bearbeiten. Ein weiterer Vorteil ist, dass bei der Deprovisionierung diese Kopie der Standard Datei ohne Nebenwirkungen gelöscht werden kann. Dadurch dass die Datei, die von den Kollegen gepflegt wird als Grundlage verwendet wird, sind alle provisionierten Instanzen immer auf dem aktuellsten Stand. Um dies Umzusetzen musste ein JCL Job geschrieben werden, der den Kopiervorgang abbildet. Anschließend wurde dieser mittels eines neuen Steps in den Workflow eingebunden. Außerdem mussten bestimmte Gruppen zu der CSD Liste der CICS Instanz hinzugefügt werden. Die JCL ist in Abbildung **??** abgebildet. Die Reihenfolge ist relevant, da es der Initialisierungsreihenfolge entspricht.

die JCL genau erkären..... bzw job im grundlagen teil erkären

Ein spezielles Augenmerk lag auf der Editierung der ‘createCICS.jcl‘-Datei. In dieser befindet sich die Definition des STC Jobs für das provisionierte CICS. Im Standard IBM Template beinhaltet diese zunächst ein Makro für die Validierung von den SIT Parametern. Noch bevor die Jobdefinition beginnt, werden alle aus der Datei für die Eingabevariablen benötigten Variablenwerte in temporäre Zwischenvariablen eingefügt. Dadurch ist eine Änderung nur an einer Stelle notwendig, falls sich etwas an der Variablen ändert. Danach folgt die Definition des Jobs, diese setzt sich aus folgenden Hauptbestandteilen zusammen:

- Einbindung der benötigten Bibliotheken
- Einbindung der zuvor angelegten CICS spezifischen Dateien
- Definition der SIT Parameter

In Abbildung **??** ist zu sehen, dass es vor allem bei der Definition der SIT Parameter zu tief verschachtelten if-Bedingungen kommen kann. Es handelt sich um den Code, der für das Einlesen der Variable ‘DFH_REGION_SITPARAMS‘ aus der Eingabedatei zuständig ist. In dieser Variablen werden die SIT Parameter als Komma separierter String angegeben. Für die Erzeugung eines DATEV eG spezifischen CICS, wurde das Makro für die Validierung von SIT Parametern beibehalten. Alles danach wurde zunächst durch eine zur Verfügung gestellten DATEV eG Standard JCL, für die Erzeugung eines CICS, ersetzt. Nach und nach ist die Logik, wie die aus Abbildung **??**, hinzugefügt worden. Zusätzlich wurden die vorher statische DATEV eG Standard JCL durch Verwendung der Templatevariablen dynamisiert.

Zu der Definition der SIT Parameter ist zu sagen, dass hier nur die wirklich benötigten mit aufgenommen wurden. Die anzunehmenden Werte wurden einzeln mit den CICS Administratorenteam besprochen und festgelegt. Es ist zu beachten, dass es im IBM Standard Template zwei Möglichkeiten gibt, die Parameter zu setzen. Für bestimmte SIT Parameter besteht eine Variable innerhalb des Templates. Für alle anderen ist die Variable

'DFH_REGION_SITPARAMS' vorgesehen. In dieser Arbeit wurde hauptsächlich auf letztere Möglichkeit gesetzt. Dadurch sind die SIT Parameter nur an einer Stelle im Template zu verwalten beziehungsweise die Verwaltung wird nicht auf zwei Arbeitsweisen verteilt.

Schließlich hat die Provisionierung eines DATEV eG spezifischen CICS Instanz funktioniert. Dies wurde mit einem Anmeldevorgang an diese CICS sichergestellt. Außerdem sind alle Standard DATEV eG Transaktionen funktionsfähig. Es wurden alle Dateien wieder pflichtgerecht gelöscht.

5.1.1.2 Bereitstellung Db2

In diesem Absatz wird die Provisionierung einer Db2 Datenbank beschrieben. Da die Systemumgebung noch der Testplex ist, nur die Datenbank ohne Tabelle, ohne Daten.

Für die Erstellung einer Db2 Datenbank existiert innerhalb der DATEV eG eine REST-API. Wie im Absatz **??** beschrieben, ist es möglich innerhalb eines Workflow Steps einen REST-Request abzusenden. Der Code ist in Abbildung **??** zu sehen. So muss im Body des Requests unter anderem der Datenbankname und eine UserID übergeben werden. Der Code für das Löschen der Datenbank sieht ähnlich aus, nur handelt es sich um einen DELETE-Request. So wurden zwei neue Steps erzeugt und in den Workflow eingebunden.

Die API ist nur dazu fähig Datenbanken auf einem bestimmten Datenbanksystem zu erzeugen. Um die Datenbank aus der CICS Instanz heraus nutzen zu können, muss dem CICS dieses Datenbanksystem mitgeteilt werden. Hierfür ist, wie in Abbildung **??** bereits zu sehen ist, das Hinzufügen einer weiteren CSD Gruppe notwendig. Des Weiteren müssen weitere Bibliotheken in der 'createCICS.jcl' aufgenommen werden. Um den Aufruf möglichst dynamisch zu gestalten wurden, zusätzlich neue Variablen im Template definiert. Diese werden in der Eingabedatei des Templates gesetzt.

In Abilldung zeile SOUNSSO

5.1.1.3 Bereitstellung MQ

In diesem Absatz wird die Provisionierung einer MQ Queue beschrieben. Es ist auch möglich einen MQ Queuemanager zu provisionieren, der Fokus dieser Arbeit liegt aber auf der Bereitstellung von Queues. Bei einem Queuemanager handelt es sich um ein Subsystem, deshalb wird vorerst von einer automatischen Bereitstellung abgesehen. Auf dem Testplex wird des Weiteren die benötigte Funktion, dass eine CICS Transaktionen über eine Queue gestartet wird, nicht untersucht. Außerdem wird zunächst nicht auf die Anforderungen der Anwendung eingegangen, sondern es wird geprüft wie es möglich ist eine einzelne Queue zu

provisionieren. Alles weitere wird erst in der Entwicklungssystemumgebung umgesetzt. Dies ist im Absatz **??** beschrieben.

Die IBM stellt für die Verwaltung und das Nutzen von Queues Programme zur Verfügung. Diese können mittels eines Jobs und bestimmten Parametern gestartet werden. In Abbildung **??** ist die JCL des Jobs für das Erstellen einer Queue zu sehen. Das auszuführende Programm ist 'CSQUTIL' und als Parameter wird der Queuemanager übergeben. Unter dem DD Namen 'MQSCIN' ist der MQ Befehl für das Erzeugen einer Queue zu sehen. Um zu Prüfen, ob die Queue auch funktionsfähig ist, wurde nach dem Erstellen auch mit Hilfe eines Jobs, eine Messages auf die Queue geschrieben und wieder abgeholt. Der Job für das Löschen der Queues ist ähnlich aufgebaut.

Ähnlich wie in Absatz **??** für die Datenbank beschrieben, muss der CSD Datei eine weitere Gruppe für den Queuemanager angegeben werden. Dadurch hat die CICS Instanz auf alle Queues, die sich innerhalb dieses Managers befinden, Zugriff. Des Weiteren ist die Aufnahme weiterer Bibliotheken in der 'createCICS.jcl' notwendig.

5.2 Entwicklungssystemumgebung

Innerhalb der Entwicklungssystemumgebung sind die Sicherheits- und Rechtevorschriften schärfer als auf dem Testplex. So wäre es zwar möglich alle für die administrativen Aufgaben notwendigen Rechte einer persönlichen UserID zu geben. Dies würde bedeuten, dass alle Anwender dieses Templates diese Rechte auch benötigen. Dies würde zu einem Chaos auf dem System führen, da sie auch außerhalb des Templates diese Rechte besitzen würden. Somit wurde in Absprache mit den Administratorenteams für CICS und MQ festgelegt hierfür jeweils einen technischen User[4] zu beantragen. Diesem werden nur die benötigten Rechte übergeben und ist somit sehr anwendungsspezifisch. Um als Anwender das Template nutzen zu können, werden nur die Rechte benötigt Jobs mit diesen technischen Usern ausführen zu dürfen. Für Db2 ist ein solcher User nicht notwendig, da das Datenbanksystem hinter der REST-API für alle zugänglich ist und jeder darauf Datenbanken erstellen darf.

Bei der Übertragung des Templates von der Testsystemumgebung auf die Entwicklungssystemumgebung waren Anpassungen in allen drei Bereichen des Templates notwendig.

5.2.1 CICS Anpassung

Zunächst wurden alle Steps modifiziert, so dass sie den CICS spezifischen technischen User verwenden. Hierfür musste der 'JOB' Baustein jeder JCL angepasst werden. Dafür bietet

[4] User ID mit zunächst keinen Berechtigungen

zOSMF beim Zuweisen des ‘Tenants‘ eine Standard Jobkarte, die vor jeden Job des Templates eingefügt wird, hinterlegt werden. Als nächstes musste ein Parameter bei der Erstellung der CICS spezifischen Dateien hinzugefügt werden. Es handelt sich um den Massageclass Parameter mit dem Wert ‘NONE’. Dadurch sind die Daten von der täglichen Datensicherung der Entwicklungssystemumgebung ausgenommen. Da die Dateien bei der Deprovisionierung gelöscht werden, ist keine Sicherung notwendig. Außerdem ändert sich die CSD Datei, die als Vorlage gilt, auf die Standard Entwicklungssystemumgebung CSD Datei. Die Db2 und MQ Bibliotheken, die das CICS anzieht, besitzen einen anderen Namen. So musste dies in der ‘createCICS.jcl‘ angepasst werden. Zusätzlich musste ein SIT Parameter angepasst werden, so dass die Log Dateien funktionisfähig sind. Außerdem kam noch ein Ordner neu hinzu. Diese dient als später als Ablageort der kompilierten Programme.

5.2.2 Db2 Anpassung

Eine genauere technische Analyse, der Rechnungsschreibungsdatenbank, kam zum Ergebnis, dass es zwar möglich wäre diese Datenbank zu provisionieren, aber es den zeitlichen Rahmen dieser Arbeit übersteigen würde. Der Grund hierfür ist die Komplexität, der benötigten Tabellen. So wird auf drei Tabellen für die Ermittlung der Produktstammdaten lesend zugegriffen, auf neun weitere bei der Bestimmung der Preisabhänigkeiten. Des Weiteren wird auf die Tabellen nicht direkt zugegriffen, sondern über Views. Dabei werden nur bei drei Tabellen eins zu eins Views verwendet. Bei allen anderen werden innerhalb der View noch weitere Tabellen, teilweise aus anderen Datenbanken, gejoint. Insgesamt besteht das System aus 14 Tabellen, die auf vier Datenbanken aufgeteilt sind, und 12 Views für den Zugriff auf diese Tabellen.

Es wurde begonnen für die drei Produktstammdatentabellen, mit den Views und den dadurch zusätzlichen Datenbanken und Tabellen, die Definition in der Data Definition Language zu erstellen. Es gibt keine Referenzen zwischen diesen Tabellen. Es stellte sich heraus, dass diese Datei aus annähernd 600 Zeilen Code besteht. Dieser ist im Anhang **??** zu finden. Dabei handelt es sich um ungetesteten Code. Nachdem dieser Code getesten und die restlichen Tabellen, Views und Datenbanken ebenfalls aufgenommen wurden, müsste das ganze System noch mit den eigentlichen Daten befüllt werden. Durch die Nutzung von Templates wäre dieser Aufwand einmalig zubewältigen, trotzdem überschreitet er diese Arbeit.

Für die weitere Arbeit werden, die in einem anderen Datenbanksystem bereits vorhanden, Datenbanken genutzt. Hierfür musste der Wert, der dafür vorgesehenen Variablen in der Eingabedatei des Templates, geändert werden. Außerdem wurden sowohl in der Provisionierungs- als auch in der Deprovisionierungsdatei die Datenbanksteps auskommentiert und somit kommen diese nicht mehr zum Einsatz.

5.2.3 MQ Anpassung

Da, wie im Absatz **??** beschrieben, sehr viele gleichartige Queues benötigt werden, wurde für die Erstellung dieser von den MQ Administratorenteam ein Rexx Skript angefertigt. Dies geschah unabhänig dieser Arbeit zum Zeitpunkt der Einführung des Rechnungsschreibungsprozesses. Dieses Skript steht dieser Arbeit zur Verfügung. Es wurde auf die Bedürfnisse des Templates angepasst.

Hierfür wurden zunächst die Eingabeparameter durch vorher angelegte Templatevariablen ersetzt. Diese steuern, wie viele Queues jeweils angelegt werden, auf welchen Queuemanager die Queues angelegt werden und den ersten Qualifier des Queuenamens. Für den restlichen Queuenamen existiert auch eine Variable, in dieser werden die Namen als Komma separierte Liste angeben und ausgelesen. An hand dieser Namen wird dann die maximale Queuetiefe und die maximale Länge einer einzelnen Nachricht festgelegt. Im alten Skript wurden die Queues mit Hilfe einer Queue, die als Vorlage dient, angelegt. Im Fall einer Provisionierung kann nicht davon ausgegangen werden, dass diese Vorlagen zur Verfügung stehen. Deshalb wurden die benötigten Parameter händisch angegeben. Um die so eben erstellten Queues zu testen, wurde eine Routine entwicklelt, die eine Nachricht auf die Queue schreibt und diese wieder abholt. Anschließend wurde es in den Provisionierungsworkflow mit Hilfe eines neuen Steps aufgenommen.

Für die Deprovisionierung besteht noch kein Skript. Als Grundlage dient das vorher angepasste Provisionierungsskript. Hierfür musste der ‘Define‘-Befehl für die Erstellung von Queues durch den ‘Delete‘-Befehl ausgetauscht werden. Die Logik für die Ermittlung der maximalen Queuetiefe und der maximalen Nachrichtenlänge wurde nicht mehr benötigt und konnte entfernt werden.

Diese beiden Skripte sind nur für den Datenaustausch zwischen der CICS Transaktion für die Preisermittlung und dem Batch Ablauf zuständig. Wie in Absatz **??** beschrieben, benötigt der Ablauf noch weitere Queues. Da es sich hierbei um spezielle Queues handelt, wurde auf die im Absatz **??** gezeigte Technik zurückgegriffen. Bei der Antwort-Queue für die Ermittlung der Listenpreise handelt es sich um eine Queue ohne besondere Parameter. Es werden noch zwei Trigger-Queues benötigt. Die über Prozesse eine Transaktion im CICS starten. Als letzter Baustein, dass das Triggering der Transaktion funktioniert, wird noch eine Initiation Queue benötigt. Diese muss im CICS hinterlegt sein.

Jeder CICS Instanz kann nur eine Initiation Queue zugewiesen sein. Dadurch benötigt jedes CICS eine eigene Initiation Queue. Die Zuweisung geschieht in der MQ CSD Gruppe. Somit müsste für jede provisionierte CICS Instanz im Voraus eine solche CSD Gruppe angelegt werden. Daraus folgt, dass in diesem Schritt beschlossen wurde, die Verwaltung der MQ CSD Gruppe komplett dem Template zu übergeben. Diese Entscheidung hatte eine Änderung

des in Abbildung **??** gezeigten Codes zur Folge. So wird, wie in Abbildung **??** abgebildet, zunächst eine Gruppe angelegt und erst anschließend dem CSD hinzugefügt.

Außerdem musste für jeden MQ bezogenen Job die Jobkarte angepasst werden, so wurden hier der technische User durch den technischen User, der für die administrativen MQ Aufgaben berechtigt ist, ausgetauscht.

5.2.4 Testablauf

Für die Prüfung der Funktionsfähigkeit der so generierten Laufzeitumgebung steht dieser Arbeit ein Testablauf zur Verfügung. Dieser wurde von den Mitarbeitern der Rechnungsschreibung beigesteuert. Dabei handelt es sich um einen Teilablauf des gesamten Rechnungsschreibungsprozesses. In diesem Ablauf wird nur die Preisermittlung, die die Laufzeitumgebung benötigt, getestet. Als Eingabe dienen vordefinierte Dateien und die Ergebnisse werden ebenfalls in Dateien geschrieben. Der Ablauf liegt in Form von zwei Jobs vor. Beide sind in der gleichen JCL Datei definiert, somit starten beide zeitgleich. Dies ist notwendig, da der erste Job die Verarbeitung im CICS über die Queues startet und der Zweite lauscht auf die Ergebnisqueues.

Um den Ablauf auch auf der vorher provisionierten Laufzeitumgebung zu starten, musste lediglich der verwendete QueueManager angepasst werden. Über die Queues und das verwendete Triggering wird die Transaktion im richtigen CICS gestartet. Bei ersten Versuch den Ablauf zu starten kam es MQ-seitig zum Fehler. Hier mussten noch zwei Queueparameter angepasst werden. Danach ist der Testablauf ohne Fehler und mit der richtigen Ausgabe gelaufen. Um die Ausgabe zu prüfen wurde der gleiche Testablauf mit den gleichen Eingabedateien auf der für Testzwecke üblichen Laufzeitumgebung durchgeführt. Anschließend wurden die Ausgabedateien beider Läufe verglichen.

5.3 Bereitstellungsprozess aktuelles Template

Betrachtet wird der Fall eins, dass das Template dem Entwicklerteam im zOSMF zur Verfügung steht. Es gibt noch keine Instanz dieses Templates. Entwickler eins möchte einen neuen Programmstand testen und benötigt somit eine Instanz des Templates. Der Mitarbeiter meldet sich an der zOSMF Oberfläche an und klickt auf den Menüleistenpunkt ‘Cloud Provisioing‘. Anschließend öftner er die ‘Software Services‘ und wählt dort das oben genannte Template aus. Er kann es ohne Änderungen provisionieren und damit seine Programmabläufe testen.

Als nächstes wird der Fall zwei dargestellt bei dem bereits eine Instanz des Templates besteht. Entwickler zwei möchte ebenfalls einen neuen Programmstand prüfen, unabhängig

von den Änderungen aus Fall eins. Somit benötigt er eine weitere Instanz des Templates. Mit dem aktuellen Stand muss er wissen an welchem Speicherort das Template abgelegt ist. Da er die Template- nicht die Workflowdateien kopieren muss. Dann sind Änderungen an den Eingabevariablen des Templates notwendig. Unter anderem ist eine andere CICS Application Id zu wählen. Das die Queues und MQ Prozesse aus Fall eins nicht überschrieben werden, muss ebenfalls ein anderer Queue Manager gesetzt werden. Anschließend muss mit den veränderten Dateien ein neues Template in zOSMF aufgenommen werden. Schließlich kann Mitarbeiter zwei eine Instanz erzeugen. Diese ist unabhängig von der Instanz aus Fall eins.

In Fall drei wird der Prozess, bei dem ein Administrator eine Änderung an einer Workflowdatei druchführt, betrachtet. Zunächst muss der Speicherort der zu bearbeiteten Dateien bekannt sein. Anschließend kann die Änderung mit einem Editor nach Wahl durchgeführt werden. Jetzt sind noch zwei weitere Fälle zu betrachten. Zunächst der Fall bei dem das Template noch nicht veröffentlicht wurde, dass heißt es steht noch keinem weiteren Team zur Verfügung. Hier kann das Template in der zOSMF Oberfläche per Mausklick aktualisiert werden. Bei dem nächsten Fall ist das Template bereits veröffentlicht. Um die Funktionsfähigkeit der veralteten Instanzen weiterhin sicherzustellen, muss eine neue Version des Templates erzeugt werden. Dies ist auch per Mausklick zu lösen.

5.4 Fazit Realisierung

Am Ende der Realisierung steht ein funktionsfähiges Template. Dieses Template provisioniert ein CICS und die benötigten MQ Queues. Wie in Absatz **??** beschrieben, wurde eine Db2 Datenbank wegen hoher Komplexität außen vorgelassen. Es hat sich aber herausgestellt, dass dies theoretisch auch möglich wäre. Zudem ist es möglich damit einen Testablauf der Rechnungsschreibung korrekt durchzuführen.

Jedoch gibt es auch Probleme. Hier sind die nicht sprechenden Fehlermeldung, Abbildung **??** als Beispiel, von zOSMF zu nennen. So wird bei dem Hinzufügen und Aktualisieren

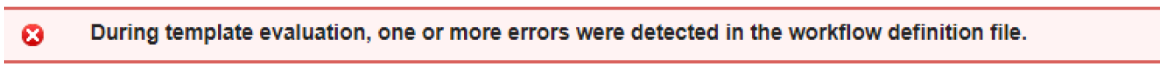

Abbildung 5.1: Beispiel einer Fehlermeldung von zOSMF

eines Templates in zOSMF das Template und damit alle davon benötigten Dateien auf Syntaxfehler geprüft. Die in Abbildung **??** gezeigte Meldung tritt dann ein, wenn ein solcher Syntaxfehler vorhanden ist. Es ist aber nicht zu erkennen, welcher Fehler genau vorliegt, noch nicht einmal in welcher Datei dieser auftritt. Zudem auch keine genaue Anzahl an auftretenden Fehlern. Dieser Umstand kombiniert mit 36 bestehenden Dateien gestaltet die

Fehlersuche zeitaufwendig. Im Gegensatz dazu wird im Fehlerfall beim Ausführen eines Steps immer der Fehlercode und der genaue Ort des Fehlers ausgegeben. Zum Beispiel wird bei einem Step, in dem ein REST Aufruf gemacht wird, und ein Fehler auftritt, der Requestcode und die hinterlegte Fehlermeldung an der zOSMF Oberfläche angezeigt.

Ein weiteres Problem ist, die verwendete Programmiersprache, die für die dynamische Generierung von Skripten sorgt. So ermöglicht diese die dynamische Wertzuweisung von zum Beispiel Rexxvariablen durch Templatevariablen. Des Weiteren besteht eine Art von String Verarbeitung. Zu beachten ist, dass wenn am Zeilenanfang ein '#' steht, kann diese Programmiersprache verwendet werden. In Abbildung **??** ist ein Beispiel zu sehen. Dort werden die Queuenamen, die als kommaseparierte Liste in der Templatevariable 'DFH_MQ_QUEUENAMES' angegeben sind, ausgelesen und in eigenen REXX Variablen gespeichert. Zusehen ist zunächst eine 'set' Anweisung mit der Variablen zugewiesen werden können. Des Weiteren sind If-Bedingungen und eine foreach-Schleife zu sehen. In Abbildung **??** wird das Ergebnis, welches zur Laufzeit ausgeführt wird, dargestellt. Es ist zu erkennen, dass nur noch die für das REXX Skript notwendigen Codeabschnitte vorhanden sind. Dadurch können sehr dynamische Templates erstellt werden. Jedoch wurde weder eine Dokumentation zu dieser Sprache noch um welche Sprache es sich genau handelt gefunden. Somit liegt dem Wissen über diese Sprache nur der Code aus Beispielen der IBM zu Grunde.

Ein weiterer Problempunkt ist das mit zOSMF und dem Template einhergehende Zugriffsrechtekonzept. Die zOSMF Berechtigungsgruppen sind nicht an die DATEV eG internen Richtlinien angepasst. Die Aufnahme in eine solche Gruppe, um zum Beispiel die zOSMF Oberfläche nutzen zu dürfen, geschieht auf Zuruf und händisches Hinzufügen einer User Id durch einen Mitarbeiter. Außerdem ist der Einsatz einer für das ganze Template gültigen Standard Jobkarte, um technische User verwenden zu können, nicht optimal. zOSMF bietet hier eigentlich eine Möglichkeit in der Stepdefinition einen 'runAsUser' anzugeben. Unter diesem User würde der Step dann ausgeführt werden, also die Stelle an der zum Beispiel für CICS Steps der technische User für administrative CICS Aufgaben angegeben werden müsste. So wird das Gewehren der expliziten Rechte zum Starten eines Jobs mit der technischen User Id eingespart. Dieses Gewehren ist wiederrum eine manuelle Arbeit, die mittels eines Formulares beantragt wird. Jedoch um einen 'runAsUser' in der Stepdefinition angeben zu können, muss in der dem Template zugewiesen 'Domain' ein sogenannter 'Cloud Security Admin' hinterlegt sein. Dieser würde sicherstellen, dass nur die für ein Template zugelassenen User diesen Template auch provisionieren dürfen. In dieser Arbeit wird die mitgelieferte 'Default Domain' genutzt, in dieser ist kein 'Cloud Security Admin' angegeben. Da es sich um die Standard 'Domain' handelt, darf diese nicht geändert werden. Somit müsste eine eigene 'Domain' angelegt werden um einen 'Cloud Security Admin' hinterlegen zu können. Dadurch dass sich zOSMF bei der DATEV eG noch in einem Teststadium befindet, wird von der Erstellung einer eigenen 'Domain' abgesehen. Dies ist der Grund für den nicht optimalen Einsatz von den oben genannten Jobkarten. An diesen beiden Gründen ist zu erkennen,

dass das Rechtekonzept noch nicht für einen firmenweiten Einsatz ausgelegt ist und noch überarbeitet und angepasst werden muss. Dies ist nicht Bestandteil dieser Arbeit.

5.5 Interviews

In diesem Absatz wird zunächst erläutert, auf welcher Grundlage die Interviews geführt worden sind. Anschließend werden die Ergebnisse der einzelnen Interviews nach Gruppen aufgeteilt, ausgewertet und interpretiert. Schließlich wird daraus ein allgemeines Stimmungsbild abgeleitet.

5.5.1 Durchführung

Interviewt wurden jeweils zwei Mitarbeiter der Gruppen, CICS Administration, Db2 Administration, MQ Administration und Entwicklerteam der Rechnungsschreibung. Zusätzlich wurde eine Fachberaterin im Bereich Technologiestrategie befragt. Sowohl der Fragenkatalog als auch die ausgefüllten und digitalisierten Fragebögen sind im Anhang **??** zu finden. Bevor die Interviews durchgeführt worden sind, wurde den Teams in getrennten Terminen die Ergebnisse dieser Arbeit vorgestellt. Der Schwerpunkt wurde an das jeweilige Team angepasst. So wurde bei den Administratorenteams vor allem auf den Teil des Templates, der für ihr Arbeitsgebiet zuständig ist, eingegangen. Außerdem wurden neben den im Absatz **??** dargestellten Lösung, auch die Lösung aus Kapitel **??** vorgestellt. An Hand des durch diese Arbeit bereit gestellte Template wurde die zOSMF Oberfläche erläutert.

5.5.1.1 CICS Administratoren

Die Meinung bezüglich des aktuell möglichen Ablauf mittels zOSMF von CICS Administrator 1 ist mittelmäßig. So bietet es zwar eine flexible Versionierung und Veröffentlichung. Jedoch ist es durch die verschiedenen Sprachen und Dokumentarten mit Startschwierigkeiten versehen. Im Gegensatz dazu sieht CICS Administrator 2 das momentane Template zumindest für CICS als ablauffähig und mehrfach einsetzbar. Den Vorteil der Konfigurierbarkeit von außerhalb des Templates durch zOSPT nennen beide Administratoren. Als Nachteil sehen sie jedoch die Notwendigkeit eines sehr dynamischen Templates und der damit verbundenen Komplexität. Die Benutzerfreundlichkeit der Oberfläche wird von CICS Administrator 1 in beiden Fällen als sehr gut betrachtet. Auf Grund dessen, dass noch nicht damit gearbeitet wurde, enthält sich CICS Administrator 2 der Bewertung. Bezüglich der Arbeitsweise bei Änderungen an den Workflow Definitionsdateien und dahinterliegenden Skripten usw., liegt die Meinung bei schlecht bis mittelmäßig. Hier fehlt beidenen eine geeignete Toolunterstützung und das damit einhergehende Syntaxhighlighting. Der erste Eindruck wird von einem hohen Ersteinrichtungsaufwand und einer zeitaufwändigen Einarbeitungsphase geprägt. Zusammen mit den verschiedenen Sprachen hat dies auf CICS Administrator 1 eine abschreckende Wirkung. Dennoch können sich beide Befragten vorstellen, nachdem der

Einarbeitungsaufwand erbracht wurde, täglich mit dem Toolkit zu arbeiten. Da bei dem aktuell etablierte Prozess ein hoher manueller Aufwand zu erbringen ist und eine Abstimmung zwischen den Administratoren und dem Entwicklerteam notwendig ist.

Zusammenfassend lässt sich sagen, dass die CICS Administratoren eine Chance im Toolkit sehen. Allerdings schreckt der hohe Einarbeitungsaufwand und die Mischung aus verschiedenen Sprachen und fehlendem Syntaxhighlighting ab.

5.5.1.2 Db2 Administratoren

Sowohl Db2 Administrator 1 als auch Db2 Administrator 2 erkennen die Möglichkeit einer Verbesserung durch zOSMF. Jedoch sind sie der Meinung, dass noch einiges an Forschung und Weiterentwicklung notwendig ist um es sinnvoll nutzen zu können. Sie stimmen auch bei ihrer Ansicht bezüglich zOSPT überein. So sehen sie den Vorteil des Kommandozeileninterfaces vorallem bei einer Endausbaustufe mit automatisiertem Deployment innerhalb einer CI/DC-Pipline und dem Einsatz von Jenkins. Db2 Administrator 2 stört sich an den Begriffen ‘Container‘ und ‘Image‘, da diese teilweise vertauscht und synonym verwendet werden. Bezüglich der Benutzerfreundlichkeit der Oberfläche fällt die Bewertung bei beiden schlecht bis mittelmäßig aus. Db2 Administrator 1 empfindet die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien als sehr schlecht, da es momentan ohne automatisches Deployment realisiert ist. Die Bewertung von Db2 Administrator 2 ist mittelmäßig, da eine Entwicklungsumgebung sinnvoll wäre, vor allem im Hinblick auf eine Syntaxprüfung. Der erste Eindruck des Toolkits ist sehr positiv. Es wird als mächtiges Tool und als Zukunft des modernen Deployments auf dem Mainframe betitelt. Jedoch wird es als sehr komplex betrachtet. Im Vergleich dazu wird der aktuell etablierte Bereitstellungsporzess ebenfalls als komplex beschrieben. Dieser funktioniere zwar sehr gut, aber es sind viele Abhänigkeiten zu anderen Personen vorhanden, dadurch entstehen Wartezeiten. Außerdem sei ein sehr umfangreiches Wissen über alle beteiligten Subsysteme notwendig. Hinzu kommt ein hoher Konfigurationsaufwand und viel Vorarbeit, zum Beispiel Funktionsuser und ein Rechtekonzept. Beide Db2 Administratoren könnten sich um diese Probleme anzugehen, vorstellen mit dem Toolkit täglich zu arbeiten. Eine Verbindung mit Jenkins wird hierfür von Db2 Administrator 1 vorausgesetzt.

Die Interviews mit den Db2 Administratoren ergaben folgendes Bild. Sie sehen in dem Toolkit eine gute Möglichkeit um den Bereitstellungsprozess zu vereinfachen und weniger zeitaufwändig zu gestalten. Allerdings ist noch viel Forschungsarbeit in dieses Thema zu investieren. Als Hauptpunkt ist die Nutzung mit Jenkins und so die Einbindung und Etablierung einer automatisierten Lösung zu nennen.

5.5.1.3 MQ Administratoren

MQ Administrator 1 sieht bereits im aktuell funktionsfähigen Template einen Mehrwert. Zum einen weil mehr Verantwortung im Entwicklerteam liegt und zum anderen sin weniger händische Eingriffe notwendig. Jedoch ist die Lösung, die im Ausblick gezeigt wurde, flexiber und damit etwas besser geeignet. Zudem seien die momentan bereits vorhanden Features durchaus gut, jedoch kam die Frage auf, ob die IBM das 'IBM Cloud Provisioning and Management for z/OS'-Toolkit noch weiterentwickelt. Die Benutzerfreundlichkeit der zOSMF Oberfläche wurde als mittelmäßig bis gut eingestuft. Bezüglich des Arbeiten, Verwalten und Ändern von Workflow Definitionsdateien und den dazugehörigen Skripten konnte keine Bewertung abgegeben werden, da noch nicht selbst damit gearbeitet wurde. Dies hat auch Einfluss auf den ersten Eindruck. So wird zu bedenken gegeben, dass der Zeitaufwand und die zu leistenden Vorarbeiten mit einzubeziehen sind. Vor allem, wenn die Provisionierung von einem MQ Queue Manager hinzu kommt. Jedoch kann sich MQ Administrator 1 vorstellen mit dem Toolkit täglich zu arbeiten, da letztenendlich die Werkzeugwahl keine Rolle spielt. Diese Entscheidung wird dadurch begünstigt, dass der aktuell etablierte Prozess schlecht beurteilt wird. Aufgrund des hohen manuellen Aufwands und der dadurch erzeugten Rückfragen. Zuletzt wird noch darauf hingewiesen, dass das Toolkit generell noch Neuland sei. So müssten erst die Grundlagen gelernt und damit Erfahrung gesammelt werden bevor eine qualitativere Bewertung möglich sei. Dies beinhaltet wahrscheinlich eine starke Lernkurve.

Im Vergleich zu MQ Administrator 1 fehlt MQ Administrator 2 noch weitere Automatismen. So sind trotz des Einsatzen von zOSPT noch Absprachen mit Dritten, wie dem RACF-Team und dem Speicher-Team, notwendig. zOSPT sei zudem nur Docker ähnlich, ist jedoch keine vollumfängliche Containerlösung. So könnte sich MQ Administrator 2 zwar vorstellen mit dem Toolkit täglich zu arbeiten, aber es müsste ohne manuelle Eingriffe funktionieren. Die Erstellung der Skripte muss mit einem einmaligen Aufwand verbunden sein, so dass sie keine ständigen Anpassungen benötigen. Davon wird auch der erste Eindruck beeinflusst. So sind zwar viele gute Ansätze vorhanden, aber es fehlen Analogien und eine Ähnlichkeit zu Jenkins und anderen PaaS Lösungen. Dies geht soweit, dass XML nicht mehr als zeitgemäß betrachtet wird, sondern auf Umsetzungen in groovy, yaml oder mit ansible playbooks zu setzen sei.

Zusammenfassend lässt sich sagen, dass sich beide MQ Administratoren einig sind, dass der momentan etablierte Bereitstellungsprozess schlecht ist und ein neuer Prozess durchaus notwendig wäre. Der durch diese Arbeit gezeigte Prozess als Ablöse wird prinzipiell als möglich erachtet. Jedoch nur der Einsatz mit zOSPT. Außerdem wird vor einer starken Lernkurve und noch fehlender Automation und der damit einhergehenden Ähnlichkeit zu Jenkins oder anderen PaaS Lösungen gewarnt.

5.5.1.4 Entwicklerteam der Rechnungsschreibung

5.5.1.5 Fachberaterin im Bereich Technologiestrategie

Nach der Fachberaterin im Bereich Technologiestrategie ist der gezeigte Ablauf beziehungsweise die zOSMF Oberfläche für eine solche Aufgabe geeignet. Jedoch sei es besser wenn zOSMF in den bereits existierenden 'Marketplace' für DATEV Cloud Lösungen integriert wäre. Der Prozess, der mit Hilfe von zOSPT ermöglicht wird, wird als gut angesehen. Da durch ihn die Entwicklung von z/OS Anwendungen an die Vorgehensweise der Cloud Native Entwicklung angenährt wird. Hier kommt die Rolle des Build Engineers auch für solche Anwendungen ins Spiel. Dieser kümmert sich um die Erstellung und Pflege der Build-Pipeline. Da sich die Fragen drei bis fünf auf die Nutzung und Verwaltung von Templates und somit auf die administrative Seite abzielen sind diese aus Entwicklersicht nicht relevant. Große Nachteile im momentan etablierten Bereitstellungsprozess ist vorallem das eine Anzahl an Entwickler, die an einem Produkt arbeiten, die gleiche Entwicklungssystemumgebung teilen. So arbeiten alle mit der gleichen CICS Instanz, der gleichen Test-Datenbank und mit den gleichen MQ Queues. Dadurch beeinflussen Änderungen die Tests der anderen Kollegen und müssen koordiniert werden. Falls Änderungen an der Umgebung notwendig sind, kann während dieser Zeit kein Entwickler weiterarbeiten. Hier sei der Vorteil des Toolkits. Es ermöglicht aus Entwicklersicht eine sehr einfache, schnelle Möglichkeit eine isolierte Umgebung bereitzustellen, unabhängig von dem Administratorenteams. Zusätzlich dienen die Konfigurationsdateien auch als Dokumentation, welche Ressourcen für ein erneutes Erstellen der Umgebung notwendig sind.

Abschließend lässt sich sagen, dass aus Sicht einer Fachberaterin im Bereich Technologiestrategie dieses Toolkit die Entwicklung beziehungsweise den Bereitstellungsprozess deutlich verbessern kann. So ist für den Entwickler ein an die Cloud Native Welt angenäherter Entwicklungsprozess möglich. Dadurch wird der Wechsel zwischen beiden Umgebungen immer fließender.

Kapitel 6

Ausblick

Je weiter das Projekt dem Abschluss näher kam desto mehr kristallisierte sich ein Hauptproblem heraus. Dieses Template ist sehr auf die Rechnungsschreibung spezialisiert, dass heißt es ist funktionsfähig, aber kann nicht ohne zeitaufwendige Eingriffe in das Template, in die Workflowdefinitionsdateien und die eigentlichen REXX Skripte und Jobs, an eine andere Anwendung angepasst werden. Zusätzlich ist der durch den momentanen Stand des Templates ermöglichte Bereitstellungsprozess nicht optimal. Bei der Betrachtung des Falles, wenn zwei Anwender jeweils eine eigene Instanz des gleichen Templates benötigen, muss dieses kopiert werden und neu in zOSMF aufgenommen werden. Dies verlangt Wissen über die zOSMF Oberfläche und den Speicherort des Templates, um es schließlich auch zu ändern. Hinzu kommt, dass die Bereitstellung eines Queue Managers nicht im Template enthalten ist. So müsste dafür ebenfalls ein neuer Queue Manager angelegt werden. Eine Herangehensweise an dieses Problem wird im Folgenden beschrieben. Diese ist als Ausblick zu verstehen, die Umsetzung ist kein Bestandteil dieser Arbeit.

Angenommen das Template beinhaltet die Provisionierung eines Queue Managers. So könnte jeder Mitarbeitern seine eigene Instanz des Templates besitzen und beispielsweise für eigene Tests nutzen. Da die Application Id einer CICS Instanz eindeutig sein muss, müsste dennoch jeder Mitarbeiter ein eigenes Template dahingehend anpassen, dass dies gewehrleistet ist. Eine Möglichkeit wäre eine Definition eines Pools mit verfügbaren Application Ids und dann mittels eines Programmes eine ungenutzte zu bestimmen. Dieses Programm kann dann in das Template mittels eines Steps aufgenommen werden. Das würde das Problem mit der eindeutigen Application Id lösen. Jedoch müsste immer noch für jede kleine Änderung an der Konfiguration des Templates ein neues erzeugt werden. Hier schafft zOSPT Abhilfe. Damit kann, wie in Absatz **??** beschrieben, mit Hilfe einer Konfigurationsdatei das Template von außerhalb gesteuert werden. Dadurch fällt das Kopieren des Template für den Mitarbeiter weg, dieser muss mittels des Kommandozeileninterfaces ein Image bauen und daraus einen Container erzeugen. Das Kommandozeileninterface hat einen weiteren Vorteil. Mit Hilfe von diesem können diese Arbeitsschritte in eine Jenkins-Ablauf aufgenommen werden. Somit läuft der Prozess automatisiert ab und nähert sich modernen Entwicklungsabläufen an.

Die Arbeitsweise und der Bereitstellungsprozess für die Laufzeitumgebung, die der Rechnungsschreibungs-Ablauf benötigt, ist dadurch vereinfacht und es werden weniger Absprachen benötigt. Allerdings ist das Template weit von einer Nutzung außerhalb der Rechnungsschreibung entfernt. Während der Realisierung dieser Arbeit wurde klar, dass anwendungsspezifische Templates nicht optimal sind und nicht den vollständigen Funktionsumfang von 'IBM Cloud Provisioning and Management for z/OS' ausreizen. So ist zu empfehlen, dass für jedes Subsystem, also CICS, Db2 und MQ, ein eigenständiges Template zu realisieren ist. Für die Realisierung davon müssten die Template dynamischer implementiert sein. Als Beispiel wird die Provisionierung von MQ Queues herangezogen. Momentan werden die Prozesse und die Trigger Queues sehr statisch angelegt. Das heißt, dass sowohl Namen als auch die damit verknüpften Queueparameter fest hinterlegt sind. Um nur ein Beispiel zu nennen. Dies müsste dahingehend angepasst werden, dass alle Parameter in einer Konfigurationsdatei angeben werden können. Es ist auch in Betracht zu ziehen, ob für den User nur bestimmte vorgefertigte Profile, wie 'klein', 'mittel' und 'groß', auswählbar sind. Dies müsste alles in dem Template implementiert werden. Hierfür ist noch viel Zeitaufwand von Seiten der Administration einzuplanen.

Angenommen es existieren jeweils ein CICS, ein Db2 und ein MQ Template und diese sind so realisiert, dass sie firmenweit eingesetzt werden können. Dann ist der nächste Schritt, die Aufnahme in den 'DATEV marketplace', möglich. Der 'DATEV marketplace' ist eine Weboberfläche mit der sich Entwicklerteams ihre benötigte PaaS-Umgebung konfigurieren können. Heute stehen ihnen dort Dienste wie MongoDB, PostgreSQL, Kafka und viele weitere zur Verfügung. In weiter Zukunft könnten hier auch Dienste wie CICS, Db2 und MQ zur Auswahl stehen. Im Hintergrund würden diese über Templates und Images Instanzen bereitstellen. Um dies zu verwirklichen könnte die von zOSMF zur Verfügung gestellte REST-API verwendet werden. Diese ermöglicht den Zugriff auf fast alle zOSMF Funktionalitäten mittels Requests. Für die 'Tenant' Zuweisung zu einem Template existiert noch kein Request. Daran ist zu erkennen, dass von Seiten von zOSMF beziehungsweise von IBM ebenfalls noch Verbesserungsmöglichkeiten bestehen.

Diese technische Umsetzung ermöglicht in Zukunft den in Diagramm **??** dargestellten Bereitstellungsprozess. Es ist zu erkennen, dass Verantwortung von den Administratorenteams an die Entwicklerteams übertragen wird. Dadurch wird Kommunikationsaufwand eingespart und einem Entwickler steht binnen weniger Minuten eine funktionsfähige Laufzeitumgebung für seine legacy z/OS Anwendung zur Verfügung. Bei Problemen oder Beratungswunsch unterstützen die Administratorenteams weiterhin.

Kapitel 7

Zusammenfassung

Zusammenfassend lässt sich sagen, dass es generell möglich ist mit dem ‘IBM Cloud Provisioning and Management for z/OS‘-Toolkit Laufzeitumgebungen für legacy z/OS Anwendungen automatisiert bereitzustellen. Das funktionsfähige Template verkürzt den Bereitstellungsprozess deutlich. Durch den Abbau von Kommunikation zwischen den Abteilungen und nur einmaligen erstellen der Skripte ist es zudem weniger fehleranfällig. Die Stakeholder sehen in diesem Template auch einen Mehrwert. Jedoch ist es noch nicht perfekt. Der Bereitstellungsprozess ist noch immer mit einigen manuellen Schritten verbunden. So muss das Template manuell kopiert und Änderungen an der Konfiguration müssen innerhalb des Templates stattfinden.

Hierfür wurde in der Arbeit eine Lösung mit Hilfe von zOSPT beleuchtet. Diese sieht in einer Endausbaustufe eine einfache Einbindung des Templates in einen automatisierten Build-Prozess, zum Beipsiel mit Jenkins, vor. Außerdem würde der Einsatz von zOSPT das Einbinden in den DATEV eG internen ‘Marketplace‘ für Cloud Lösungen ermöglichen. Um diese Ziele zu erreichen muss noch viel Aufwand in die Gestaltung des Templates gesteckt werden. Zusätzlich müsste ein sogenannter ‘Service Broker‘ für die Einbindung der einzelnen Subsysteme in den ‘Marketplace‘ implementiert werden. Diese beiden Lösungsansätze stoßen sowohl bei den Administratorenteams als auch beim involvierten Entwicklerteam auf fruchtbaren Boden. Dadurch wird eine Ähnlichkeit zum ‘Cloud Native‘-Bereitstellungsprozesses hergestellt. Dies ist ein weiterer Schritt um dem Image eines veralteten Systems mit veralteten langsamen Prozesses zu enkommen.

Anhang A

Anhang

A.1 Produktstammdaten data definition language

```
1  CREATE TABLE PSSSCHEMA.TPAUSPSSBASELBART
2          (PARTITIONID INTEGER
3                                                          NOT NULL
4  WITH DEFAULT 1
5          ,PID INTEGER
6                                                          NOT NULL
7          ,AUSPRAEGUNG SMALLINT
8                                                          NOT NULL
9          ,GUELTIGAB DATE
10                                                         NOT NULL
11         ,LFDNR INTEGER
12                                                         NOT NULL
13 WITH DEFAULT
14         ,GUELTIGBIS DATE
15                                                         NOT NULL
16 WITH DEFAULT "9999-12-31"
17         ,PSSID INTEGER
18 WITH DEFAULT NULL
19         ,ZUSATZID CHARACTER(4) FOR SBCS DATA
20 WITH DEFAULT NULL
21         ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
22 WITH DEFAULT NULL
23         ,MWSTANTEILFREI DECIMAL(5, 2)
24 WITH DEFAULT NULL
25         ,MWSTANTEILREDUZIERT DECIMAL(5, 2)
26 WITH DEFAULT NULL
27         ,MWSTANTEILVOLL DECIMAL(5, 2)
28 WITH DEFAULT NULL
```

```
29          ,CONSTRAINT PID PRIMARY KEY
30          (PARTITIONID
31          ,PID
32          ,AUSPRAEGUNG
33          ,GUELTIGAB
34          ,LFDNR
35                )
36        )
37        IN DATABASE PSSBAPRV
38  APPEND NO
39  NOT VOLATILE CARDINALITY
40  DATA CAPTURE NONE
41  AUDIT NONE
42  CCSID EBCDIC
43  PARTITION BY RANGE
44        (PARTITIONID NULLS LAST ASC
45        )
46        ( PARTITION 1
47            ENDING ( 1
48        ) INCLUSIVE
49        , PARTITION 2
50            ENDING ( 2
51        ) INCLUSIVE
52        );
53
54  CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSBASELBART
55            ON PSSSCHEMA.TPAUSPSSBASELBART
56          (PARTITIONID ASC
57          ,PID ASC
58          ,AUSPRAEGUNG ASC
59          ,GUELTIGAB ASC
60          ,LFDNR ASC
61          )
62           INCLUDE NULL KEYS
63           CLUSTER
64           PARTITIONED
65           DEFINE YES
66           COMPRESS NO
67           BUFFERPOOL BP2
68           CLOSE YES
```

```
69          DEFER NO
70          COPY NO
71          PARTITION BY RANGE
72         (PARTITION 1
73                     USING STOGROUP STAPSA01
74                                    PRIQTY -1
75                                    SECQTY -1
76                                    ERASE NO
77                     FREEPAGE 0
78                     PCTFREE 10
79                     GBPCACHE CHANGED
80         ,PARTITION 2
81                     USING STOGROUP STAPSA01
82                                    PRIQTY -1
83                                    SECQTY -1
84                                    ERASE NO
85                     FREEPAGE 0
86                     PCTFREE 10
87                     GBPCACHE CHANGED);
88
89    CREATE TABLE PSSSCHEMA.TMAXNUM
90          (MAXNUMID INTEGER
91                                                          NOT NULL
92          ,MAXNUM INTEGER
93                                                          NOT NULL
94          ,MAXNUBEZ CHARACTER(42) FOR SBCS DATA
95                                                          NOT NULL
96  WITH DEFAULT "X"
97          ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
98                                                          NOT NULL
99  WITH DEFAULT "X"
100         ,CONSTRAINT MAXNUMID PRIMARY KEY
101         (MAXNUMID
102         )
103         )
104        IN DATABASE PSSBAPRV
105  APPEND NO
106  NOT VOLATILE CARDINALITY
107  DATA CAPTURE NONE
108  AUDIT NONE
```

```
109  CCSID EBCDIC;
110
111      COMMENT ON TABLE PSSSCHEMA.TMAXNUM
112           IS "maximale Nummer";
113
114
115 SET CURRENT SQLID = "DB2SADM";
116
117
118     COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUMID
119           IS "ID für maximalen Nummer";
120
121
122 SET CURRENT SQLID = "DB2SADM";
123
124
125     COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUM
126           IS "maximale Nummer";
127
128
129 SET CURRENT SQLID = "DB2SADM";
130
131
132     COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUBEZ
133           IS "Bezeichnung für maximale Nummer";
134
135
136 SET CURRENT SQLID = "DB2SADM";
137
138
139     COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.BEZEICHNUNG
140           IS "Bezeichnung für maximale Nummer";
141
142 CREATE UNIQUE INDEX PSSSCHEMA.PMAXNUM
143          ON PSSSCHEMA.TMAXNUM
144         (MAXNUMID ASC
145         )
146          INCLUDE NULL KEYS
147          CLUSTER
148          DEFINE YES
```

```
149           COMPRESS NO
150           BUFFERPOOL BP2
151           CLOSE YES
152           DEFER NO
153           COPY NO
154           USING STOGROUP STALDL01
155                 PRIQTY -1
156                 SECQTY -1
157                 ERASE NO
158           FREEPAGE 0
159           PCTFREE 99
160           GBPCACHE CHANGED
161           PIECESIZE 2097152K;
162
163   CREATE FUNCTION PSS.WHICH_PARTITIONID
164      (
165       MAXID INTEGER )
166     RETURNS INTEGER
167     VERSION V1
168     DISALLOW DEBUG MODE
169     ASUTIME NO LIMIT
170     INHERIT SPECIAL REGISTERS
171     WLM ENVIRONMENT FOR DEBUG MODE DB0TWLM
172     APPLICATION ENCODING SCHEME EBCDIC
173     QUALIFIER UGPSENT
174     DYNAMICRULES RUN
175     WITH EXPLAIN
176     WITHOUT IMMEDIATE WRITE
177     ISOLATION LEVEL UR
178     OPTHINT ""
179     REOPT NONE
180     VALIDATE RUN
181     ROUNDING DEC_ROUND_HALF_EVEN
182     DATE FORMAT ISO
183     DECIMAL( 31 )
184     FOR UPDATE CLAUSE REQUIRED
185     TIME FORMAT ISO
186     CURRENT DATA NO
187     DEGREE 1
188     PACKAGE OWNER UGPSENT
```

```
189    BUSINESS_TIME SENSITIVE NO
190    SYSTEM_TIME SENSITIVE NO
191    ARCHIVE SENSITIVE NO
192    APPLCOMPAT V10R1
193    LANGUAGE SQL
194    NO EXTERNAL ACTION
195    PARAMETER CCSID EBCDIC
196    DETERMINISTIC
197             NOT SECURED
198           CALLED ON NULL INPUT
199           READS SQL DATA
200           SPECIFIC WHICH_PARTITIONID
201  BEGIN
202            DECLARE MAXNUM INTEGER;
203            SELECT MAXNUM
204                   INTO MAXNUM
205              FROM AVADMIN.AMAXNUM
206             WHERE MAXNUMID = MAXID;
207            RETURN MAXNUM;
208  END;
209
210  SET PATH = "PSS","SYSIBM","SYSFUN","SYSPROC","SYSIBMADM","PSSSCHEMA";
211
212  CREATE VIEW PSSSCHEMA.VPAUSPSS_BASELBART
213              ( PARTITIONID
214              , PID
215              , AUSPRAEGUNG
216              , GUELTIGAB
217              , LFDNR
218              , GUELTIGBIS
219              , PSSID
220              , ZUSATZID
221              , BEZEICHNUNG
222              , MWSTANTEILFREI
223              , MWSTANTEILREDUZIERT
224              , MWSTANTEILVOLL
225              ) AS
226  SELECT B.* FROM TPAUSPSSBASELBART B WHERE B.PARTITIONID =
227    PSS.WHICH_PARTITIONID ( 3011 )
228  ;
```

```
229
230   CREATE TABLE PSSSCHEMA.TPAUSPSSPREISE
231         (PARTITIONID INTEGER
232                                                          NOT NULL
233 WITH DEFAULT 1
234         ,ARTNR INTEGER
235                                                          NOT NULL
236         ,PREISTYPID SMALLINT
237                                                          NOT NULL
238         ,GUELTIGAB DATE
239                                                          NOT NULL
240         ,STAFFELNR INTEGER
241                                                          NOT NULL
242         ,PID INTEGER
243                                                          NOT NULL
244 WITH DEFAULT
245         ,PREISREGEL CHARACTER(2) FOR SBCS DATA
246                                                          NOT NULL
247 WITH DEFAULT "X"
248         ,GUELTIGBIS DATE
249                                                          NOT NULL
250 WITH DEFAULT "9999-12-31"
251         ,PRODUKTPREIS DECIMAL(11, 3)
252 WITH DEFAULT NULL
253         ,EINZELPREIS DECIMAL(8, 3)
254 WITH DEFAULT NULL
255         ,PREISINTERVALL DECIMAL(11, 3)
256 WITH DEFAULT NULL
257         ,PREISEINHEIT DECIMAL(8, 3)
258 WITH DEFAULT NULL
259         ,STAFFELVERTEILUNG CHARACTER(1) FOR SBCS DATA
260 WITH DEFAULT NULL
261         ,INTERVALLVON INTEGER
262 WITH DEFAULT NULL
263         ,INTERVALLBIS INTEGER
264 WITH DEFAULT NULL
265         ,PREISTYPBEZ VARCHAR(100) FOR SBCS DATA
266                                                          NOT NULL
267 WITH DEFAULT "X"
268         ,PREISAB DECIMAL(8, 3)
```

```
269 WITH DEFAULT NULL
270         ,PREISABRELEVANZ CHARACTER(1) FOR SBCS DATA
271                                                 NOT NULL
272 WITH DEFAULT "K"
273         ,EINHEIT INTEGER
274 WITH DEFAULT NULL
275         ,CONSTRAINT PPAUSPSSPREISE PRIMARY KEY
276         (PARTITIONID
277         ,ARTNR
278         ,PREISTYPID
279         ,GUELTIGAB
280         ,STAFFELNR
281         )
282         )
283         IN DATABASE PSSBAPRV
284  APPEND NO
285  NOT VOLATILE CARDINALITY
286  DATA CAPTURE NONE
287  AUDIT NONE
288  CCSID EBCDIC;
289
290  CREATE INDEX PSSSCHEMA.IPAUSPSSPREISE
291          ON PSSSCHEMA.TPAUSPSSPREISE
292         (PARTITIONID ASC
293         ,PID ASC
294         )
295          INCLUDE NULL KEYS
296          NOT CLUSTER
297          DEFINE YES
298          COMPRESS NO
299          BUFFERPOOL BP2
300          CLOSE YES
301          DEFER NO
302          COPY NO
303          USING STOGROUP STAPSA01
304               PRIQTY -1
305               SECQTY -1
306               ERASE NO
307          FREEPAGE 0
308          PCTFREE 10
```

```
309          GBPCACHE CHANGED
310          PIECESIZE 2097152K;
311
312 CREATE INDEX PSSSCHEMA.IPAUSPSSPREISE2
313          ON PSSSCHEMA.TPAUSPSSPREISE
314         (PARTITIONID ASC
315         ,ARTNR ASC
316         ,PREISTYPID ASC
317         ,GUELTIGAB ASC
318         ,INTERVALLVON ASC
319         )
320          INCLUDE NULL KEYS
321          NOT CLUSTER
322          DEFINE YES
323          COMPRESS NO
324          BUFFERPOOL BP2
325          CLOSE YES
326          DEFER NO
327          COPY NO
328          USING STOGROUP STAPSA01
329               PRIQTY -1
330               SECQTY -1
331               ERASE NO
332          FREEPAGE 0
333          PCTFREE 10
334          GBPCACHE CHANGED
335          PIECESIZE 2097152K;
336
337 CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSPREISE
338          ON PSSSCHEMA.TPAUSPSSPREISE
339         (PARTITIONID ASC
340         ,ARTNR ASC
341         ,PREISTYPID ASC
342         ,GUELTIGAB ASC
343         ,STAFFELNR ASC
344         )
345          INCLUDE NULL KEYS
346          CLUSTER
347          DEFINE YES
348          COMPRESS NO
```

```
349         BUFFERPOOL BP2
350         CLOSE YES
351         DEFER NO
352         COPY NO
353         USING STOGROUP STAPSA01
354               PRIQTY -1
355               SECQTY -1
356               ERASE NO
357         FREEPAGE 0
358         PCTFREE 10
359         GBPCACHE CHANGED
360         PIECESIZE 2097152K;
361
362   CREATE TABLE PSSSCHEMA.TPAUSPSSBART
363        (PARTITIONID INTEGER
364                                                          NOT NULL
365 WITH DEFAULT 1
366         ,PID INTEGER
367                                                          NOT NULL
368         ,ARTNR INTEGER
369                                                          NOT NULL
370 WITH DEFAULT
371         ,ANDAT DATE
372                                                          NOT NULL
373 WITH DEFAULT "1966-02-14"
374         ,OPDATBEN CHARACTER(1) FOR SBCS DATA
375                                                          NOT NULL
376 WITH DEFAULT "J"
377         ,AUSDIENSTREL CHARACTER(1) FOR SBCS DATA
378                                                          NOT NULL
379 WITH DEFAULT "N"
380         ,VERTRIEBSREL CHARACTER(1) FOR SBCS DATA
381                                                          NOT NULL
382 WITH DEFAULT "N"
383         ,VERTRELDAT DATE
384 WITH DEFAULT NULL
385         ,KOMMASTELLEN INTEGER
386 WITH DEFAULT NULL
387         ,ERTRNR INTEGER
388                                                          NOT NULL
```

```
389 WITH DEFAULT
390         ,GFEDNR INTEGER
391                                                         NOT NULL
392 WITH DEFAULT
393         ,UPLONR INTEGER
394                                                         NOT NULL
395 WITH DEFAULT
396         ,MWSTEUERSATZ INTEGER
397 WITH DEFAULT NULL
398         ,POLINR INTEGER
399                                                         NOT NULL
400 WITH DEFAULT
401         ,EXPGNR INTEGER
402                                                         NOT NULL
403 WITH DEFAULT
404         ,EXPONR INTEGER
405                                                         NOT NULL
406 WITH DEFAULT
407         ,ARTIKELTYPID INTEGER
408                                                         NOT NULL
409 WITH DEFAULT
410         ,ARTIKELTYPALT CHARACTER(4) FOR SBCS DATA
411                                                         NOT NULL
412 WITH DEFAULT "0000"
413         ,BERBESTEINHID INTEGER
414                                                         NOT NULL
415 WITH DEFAULT
416         ,BERBESTEINHALT CHARACTER(4) FOR SBCS DATA
417                                                         NOT NULL
418 WITH DEFAULT "0000"
419         ,LEISTGRUPID INTEGER
420                                                         NOT NULL
421 WITH DEFAULT
422         ,LEISTGRUPALT CHARACTER(4) FOR SBCS DATA
423                                                         NOT NULL
424 WITH DEFAULT "0000"
425         ,BERFREQID INTEGER
426                                                         NOT NULL
427 WITH DEFAULT
428         ,NUTZERID INTEGER
```

```
429 WITH DEFAULT
430         ,LEISTARTID INTEGER
431                                                         NOT NULL
432 WITH DEFAULT
433         ,BERFREQALT CHARACTER(4) FOR SBCS DATA
434                                                         NOT NULL
435 WITH DEFAULT "0000"
436         ,LEISTARTALT CHARACTER(4) FOR SBCS DATA
437                                                         NOT NULL
438 WITH DEFAULT "99"
439         ,NUTZERALT CHARACTER(1) FOR SBCS DATA
440                                                         NOT NULL
441 WITH DEFAULT "K"
442         ,BARTBEZ_20 CHARACTER(20) FOR SBCS DATA
443                                                         NOT NULL
444 WITH DEFAULT "X"
445         ,ARTIKELTYPBEZ CHARACTER(50) FOR SBCS DATA
446                                                         NOT NULL
447 WITH DEFAULT "Keine␣Zuordnung"
448         ,BERBESTEINHBEZ CHARACTER(50) FOR SBCS DATA
449                                                         NOT NULL
450 WITH DEFAULT "Keine␣Zuordnung"
451         ,LEISTGRUPBEZ CHARACTER(50) FOR SBCS DATA
452                                                         NOT NULL
453 WITH DEFAULT "Keine␣Zuordnung"
454         ,BERFREQBEZ CHARACTER(50) FOR SBCS DATA
455                                                         NOT NULL
456 WITH DEFAULT "Keine␣Zuordnung"
457         ,NUTZERBEZ CHARACTER(50) FOR SBCS DATA
458                                                         NOT NULL
459 WITH DEFAULT "Keine␣Zuordnung"
460         ,LEISTARTBEZ CHARACTER(50) FOR SBCS DATA
461                                                         NOT NULL
462 WITH DEFAULT "Keine␣Zuordnung"
463         ,BARTBEZ_100 VARCHAR(100) FOR SBCS DATA
464                                                         NOT NULL
465 WITH DEFAULT "X"
466         ,ERTRBEZ VARCHAR(100) FOR SBCS DATA
467                                                         NOT NULL
468 WITH DEFAULT "X"
```

```
469         ,GFEDBEZ VARCHAR(100) FOR SBCS DATA
470                                                    NOT NULL
471 WITH DEFAULT "X"
472         ,UPLOBEZ VARCHAR(100) FOR SBCS DATA
473                                                    NOT NULL
474 WITH DEFAULT "X"
475         ,POLIBEZ VARCHAR(100) FOR SBCS DATA
476                                                    NOT NULL
477 WITH DEFAULT "X"
478         ,EXPGBEZ VARCHAR(100) FOR SBCS DATA
479                                                    NOT NULL
480 WITH DEFAULT "X"
481         ,EXPOBEZ VARCHAR(100) FOR SBCS DATA
482                                                    NOT NULL
483 WITH DEFAULT "X"
484         ,HAKONR INTEGER
485 WITH DEFAULT NULL
486         ,HAKOBEZ VARCHAR(100) FOR SBCS DATA
487 WITH DEFAULT NULL
488         ,INPGNR INTEGER
489                                                    NOT NULL
490 WITH DEFAULT
491         ,INPGBEZ VARCHAR(100) FOR SBCS DATA
492                                                    NOT NULL
493 WITH DEFAULT "X"
494         ,BEZ035 CHARACTER(35) FOR SBCS DATA
495                                                    NOT NULL
496 WITH DEFAULT "X"
497         ,CONSTRAINT PPAUSPSSBART PRIMARY KEY
498         (PARTITIONID
499         ,PID
500         )
501         ,CONSTRAINT UPAUSPSSBART UNIQUE
502         (PARTITIONID
503         ,ARTNR
504         )
505         )
506         IN DATABASE PSSBAPRV
507  APPEND NO
508  NOT VOLATILE CARDINALITY
```

```
509  DATA CAPTURE NONE
510  AUDIT NONE
511  CCSID EBCDIC
512  PARTITION BY RANGE
513         (PARTITIONID NULLS LAST ASC
514         )
515         ( PARTITION 1
516             ENDING ( 1
517         ) INCLUSIVE
518         , PARTITION 2
519             ENDING ( 2
520         ) INCLUSIVE
521         );
522
523 CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSBART
524          ON PSSSCHEMA.TPAUSPSSBART
525         (PARTITIONID ASC
526         ,PID ASC
527         )
528          INCLUDE NULL KEYS
529          CLUSTER
530          PARTITIONED
531          DEFINE YES
532          COMPRESS NO
533          BUFFERPOOL BP2
534          CLOSE YES
535          DEFER NO
536          COPY NO
537          PARTITION BY RANGE
538         (PARTITION 1
539                     USING STOGROUP STAPSA01
540                                    PRIQTY -1
541                                    SECQTY -1
542                                    ERASE NO
543                     FREEPAGE 0
544                     PCTFREE 10
545                     GBPCACHE CHANGED
546         ,PARTITION 2
547                     USING STOGROUP STAPSA01
548                                    PRIQTY -1
```

```
549                                         SECQTY -1
550                                         ERASE NO
551                         FREEPAGE 0
552                         PCTFREE 10
553                         GBPCACHE CHANGED);
554
555 CREATE UNIQUE INDEX PSSSCHEMA.UPAUSPSSBART
556           ON PSSSCHEMA.TPAUSPSSBART
557          (PARTITIONID ASC
558          ,ARTNR ASC
559          )
560           INCLUDE NULL KEYS
561           NOT CLUSTER
562           DEFINE YES
563           COMPRESS NO
564           BUFFERPOOL BP2
565           CLOSE YES
566           DEFER NO
567           COPY NO
568           USING STOGROUP STAPSA01
569                PRIQTY -1
570                SECQTY -1
571                ERASE NO
572           FREEPAGE 0
573           PCTFREE 10
574           GBPCACHE CHANGED
575           PIECESIZE 2097152K;
576
577 CREATE UNIQUE INDEX PSSSCHEMA.UPAUSPSSBART
578           ON PSSSCHEMA.TPAUSPSSBART
579          (PARTITIONID ASC
580          ,ARTNR ASC
581          )
582           INCLUDE NULL KEYS
583           NOT CLUSTER
584           DEFINE YES
585           COMPRESS NO
586           BUFFERPOOL BP2
587           CLOSE YES
588           DEFER NO
```

```
589         COPY NO
590         USING STOGROUP STAPSA01
591               PRIQTY -1
592               SECQTY -1
593               ERASE NO
594         FREEPAGE 0
595         PCTFREE 10
596         GBPCACHE CHANGED
597         PIECESIZE 2097152K;
```

A.2 Interview FragebÃűgen

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis