



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

© 2020

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

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 beigelegt. 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 [[Kuhn 19](#)] Zitattest 2 [[Roge 11](#)]

Abstract

english translation of ‘kurzzusammenfassung‘

Inhaltsverzeichnis

1. Einleitung	1
2. Grundlagen	5
2.1. Mainframe / Großrechner	5
2.1.1. Batch	5
2.1.2. Batch-Job / Job	5
2.2. Customer Information Control System	6
2.2.1. CICS Transaktion	6
2.2.2. Voraussetzungen	7
2.2.3. Einrichtung CICS Instanz	7
2.2.4. Entfernung CICS Instanz	8
2.3. IBM MQ	8
2.3.1. Queue Manager	8
2.3.2. Queues	9
2.3.3. Process	9
2.4. IBM Cloud Provisioning and Management for z/OS	9
2.4.1. Begriffserklärung	9
2.4.2. IBM Cloud Provisioning and Management for z/OS	11
3. Vorgehensweise	15
4. Analyse	17
4.1. Rechnungsschreibung	17
4.1.1. Beschreibung	17
4.1.2. Architektur der Preisermittlung	20
4.2. Aktueller Bereitstellungsprozess	21
5. Realisierung	23
5.1. Testplex	23
5.1.1. IBM Standard CICS Template	23
5.2. Entwicklungsumgebung	28
5.2.1. CICS Anpassung	28
5.2.2. Db2 Anpassung	29
5.2.3. MQ Anpassung	29

5.2.4. Testablauf	31
5.3. Bereitstellungsprozess aktuelles Template	31
5.4. Fazit Realisierung	32
5.5. Interviews	34
6. Ausblick	35
7. Zusammenfassung	37
A. Anhang	39
A.1. Produktstammdaten data definition language	39
Abbildungsverzeichnis	55
Tabellenverzeichnis	57
Quellcodeverzeichnis	59
Literaturverzeichnis	61

Kapitel 1.

Einleitung

Vor mehr als fünfzig Jahren wurde der allererste Großrechner, auch Mainframe genannt, vorgestellt. Seit dieser Zeit setzen sich die monolithisch aufgebauten Systeme in Bezug auf Leistungsfähigkeit und Zuverlässigkeit gegenüber andere Systeme ab. Obwohl die Systeme immer weniger Platz brauchten, anfangs waren es ganze Gebäudestockwerke, heute sind es ungefähr die Ausmaße eines großen Kleiderschranks. Und weiteren Verbesserung bei der Handhabbarkeit, von reinen Druckausgaben über text-basierenden Terminals bis hin zu benutzerfreundlichen GUI's. Auch hat sich die Weise, wie Programme entwickelt werden verändert. Zu Beginn mussten diese noch auf Lochkarten (ABBILDUNG !!) gestanzt und umständlich über ein Lesegerät eingelesen werden. Heute stehen dem Entwickler moderne IDE's zur Verfügung. Trotz dieser Veränderungen verlor der Mainframe durch die Dezentralisierung der IT hinzu Client-Server-Umgebungen in den 1990-er Jahren an Bedeutung. Dieser Prozess führte soweit, dass in den frühen 1990-er Jahren bereits Vorhersagen über die Abschaltung des letzten Mainframes getroffen wurden.¹ [Ceru 03]

Trotz dieser Vorhersagen verarbeiten heutzutage Großrechner weltweit circa 1,2 Millionen CICS² Transaktionen pro Sekunde.³ Im Vergleich hierzu werden 63.000 Google Suchanfragen pro Sekunde abgesetzt.⁴ Wie hat es diese schon seit den frühen 1990-er Jahren totgesagte Technologie geschafft auch heute noch diese Relevanz zu haben? Hier kommen die klar definierten Vorteile und Use-Cases des Mainframes zum Tragen. Zunächst ist 'RAS'⁵ zu nennen. Dies beschreibt grundsätzlich, die Stabilität eines Hard- und Softwaresystems. Hierzu zählt vor allem das Verhalten bei einem Hardware-/Softwaredefekt und möglichst automatische Erkennung und möglichst effektive Behebung von diesen. Zusätzlich sollte dies keinen oder nur selten einen kompletten Systemausfall zur Folge haben. Abbildung 1.1 zeigt die ungeplante Server Ausfallzeit in Minuten pro Server im Jahr 2019. Wie zu sehen ist, schneidet der IBM z Systems w/Linux oder z/OS, das Mainframesystem der IBM, am besten ab.

¹[Also 93]

²Begriffserklärung zu CICS in 2.2

³[IBM 19b]

⁴[Sull 16]

⁵reliability, availability and serviceability



Abbildung 1.1.: Annual amount of unplanned server downtime worldwide in 2019, by hardware platform

Hinzu kommen spezielle Sicherheitsmechanismen und Skalierbarkeit. All dies verbunden mit der durch (HIER SPECS EINFÜGEN) gewährleisteten Performance, ermöglicht spezielle Use-Cases. Unter anderem Massendatenverarbeitung, die dazugehörige Ressourcenverwaltung und Breitband Kommunikation. Das macht den Mainframe vor allem für Banken, das Gesundheitswesen, Versicherungen, Fluggesellschaften usw. attraktiv. Zu diesen Unternehmen zählt auch die DATEV eG. [IBM 14]

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.[DATE 17] Heute umfasst das Leistungsspektrum der DATEV eG unter anderen das Rechnungswesen, Personalwirtschaft, Consulting, IT-Sicherheit, Weiter-

bildung. Ein nicht unbeträchtlicher Teil (PROZENTSATZ ?) der betriebswirtschaftlichen Anwendungen laufen bis heute auf einem IBM Großrechner. So werden pro Tag circa 150.000 Batch Jobs und circa 90 Millionen CICS-Transaktionen verarbeitet. Hierfür stehen dem System 114.000 MIPS an CPU-Kapazität zur Verfügung. Diese Last wird von circa 14.000 aktiven Modulen erzeugt. Wie in der Abbildung ?? zu sehen ist, ist COBOL mit XXX Prozent die am häufigsten verwendete Programmiersprache am Großrechner bei der DATEV eG. Durch diese Module werden unter anderem im Monat circa 13 Millionen Lohnabrechnungen erstellt und circa 1 Millionen Umsatzsteuer-Voranmeldungen durchgeführt.

Die Risiken, die sich für die DATEV eG durch die Nutzung eines IBM Großrechners ergeben, werden im Folgenden dargestellt.

Zunächst ist zu nennen, dass die Verfügbarkeit von Skills im Mainframebereich immer schlechter wird. Die aktuellen Wissensträger fallen durch ihr Alter langsam aus. Durch die geringe Beliebtheit und wenig Präsenz an Universitäten und Hochschulen sind junge Nachfolger nur schwer zu finden. So ist zum Beispiel die Programmiersprache COBOL auf den TIOBE Index Platz 28.⁶ Zum anderen gibt es in Deutschland nur XXX Universitäten und Hochschulen, die einen Studiengang mit Schwerpunkt Mainframe anbieten.[?]

Als nächstes ist die Herstellerabhängigkeit von IBM zu erwähnen. Die DATEV eG ist nicht nur in der Wahl des Betriebssystems eingeschränkt, sondern auch einem Datenbanksystem oder einer Messaging Lösung. Außerdem hat die IBM eine Quasi-Monopolstellung[?] im Mainframebereich, so ist die DATEV eG auch in ihrer Preisverhandlungspolitik eingeschränkt. Hinzu kommt die Abhängigkeit von der IBM Mainframe Strategie, also ob die IBM selbst noch weitere Ressourcen in ihren Großrechnerbereich investiert. Dies wird durch eine sinkende Kundenzahl am Markt verstärkt.⁷

Für die DATEV eG ist trotz der Risiken eine Ablösung der Mainframe Bestandsanwendungen durch cloud-native Anwendungen aktuell nicht absehbar. Ein weiterer Punkt ist, dass die Sicherung des Bestandsgeschäfts über einen Zeitraum von Jahren oder Jahrzehnten notwendig sein wird. Aufgrund der Stärken des Großrechners will man bei Modernisierungsprojekten eine Alternative für die cloud-native Entwicklung anbieten. Um dieses Ziel zu erreichen, wurde bei der DATEV eG in den letzten Jahren der Entwicklungsprozess für Mainframe-Projekte überarbeitet und diesen an übliche Entwicklungsstandards anzupassen. So wird auf eine von der IBM entwickelten auf eclipse-basierende Entwicklungsumgebung gesetzt. Somit ist die DATEV eG auch in diesem Bereich von der IBM abhängig. Außerdem läuft seit einiger Zeit die Umstellung auf git für die Verwaltung von COBOL und IBM-Assembler Sourcen.

Zum Entwicklungsprozess zählt jedoch nicht nur das Erzeugen von Code, sondern auch die Bereitstellung der dazugehörigen Laufzeitumgebung. So gewinnen bei der DATEV eG,

⁶[[TIOB 19](#)]

⁷weltweite installierte MIPS-Zahl sei laut IBM steigend

außerhalb des Mainframeumfelds, PaaS (Plattform as a Service) Ansätze immer mehr an Bedeutung. Ein Vorteil dabei ist die unkomplizierte, automatisierte Provisionierung von Laufzeitumgebungen. Dadurch wird die Entwicklungsgeschwindigkeit erhöht und die Bereitstellung von isolierten Testumgebungen vereinfacht. Außerdem können während eines laufenden Entwicklungsprozesses Komponenten, wie zum Beispiel ein Datenbanksystem, hinzugefügt oder ausgetauscht werden. Um sich auch in diesem Bereich der großrechnerfremden Entwicklung anzupassen, sucht die DATEV eG hier nach einer Lösung. Für die automatisierte Provisionierung von Laufzeitumgebungen im Mainframeumfeld stellt die IBM seit dem Jahr 2019 das Tool ‘IBM Cloud Provisioning and Management for z/OS’⁸ zur Verfügung. Anhand des Beispiels der Rechnungsschreibung⁹ wird in dieser Arbeit untersucht, ob und wie es möglich ist, ähnlich wie bei einem PaaS Ansatz, solche Provisionierungsmechanismen für z/OS Anwendungen zu automatisieren.

⁸Beschreibung in Absatz 2.4

⁹Beschreibung im Kapitel 4.1.1

Kapitel 2.

Grundlagen

In diesem Kapitel werden für diese Arbeit wichtige Begriffe und Systeme 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. So wird er von Unternehmen verwendet, um dort kommerzielle Datenbanken, Transaktionsserver und Anwendungen, die einen hohen Grad an Sicherheit und Verfügbarkeit benötigten, zu hosten. Im Gegensatz zu verteilten Serversystemen, bei dem 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. [\[Ebbe 11\]](#)

2.1.1. Batch

Batch beziehungsweise Batch-Verarbeitung ist eine Art einer Stapelverarbeitung. Das heißt, dass Programme mit minimalem menschlichen Eingreifen nacheinander abgearbeitet werden. Dies geschieht meist zu einer vorher festgelegten Zeit. Zum Beispiel wird einmal am Tag zu einer ganz bestimmten Uhrzeit die Tägliche Bewertung der Rechnungsschreibung durchgeführt. Die auszuführenden Programme sind in sogenannten Batch-Jobs definiert. [\[Ebbe 11\]](#)

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 Eingaben und Ausgaben gestartet werden soll. Dafür wird die ‘Job Control Language’, kurz JCL, genutzt. Die drei Grundbausteine, die JCL bereitstellt, werden im Folgendem beschrieben.

Zunächst ist ‘JOB’ zu nennen. 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 jedes als einzelner sogenannter ‘Job step’ bezeichnet. Dabei können dem Programm neben den Eingabedateien auch weitere Parameter übergeben werden.

Als letztes ist der ‘DD’ Baustein zu nennen. ‘DD’ steht für Data Definition. Es verknüpft den Namen des Bausteins zum Beispiel an eine Datei, an ein I/O Gerät oder Funktionen, die im eigentlichen Programm enthalten sind. So steuert es die Ein- und Ausgaben des Programmes. Ein ‘DD’ Baustein ist immer an ein ‘EXEC’ Befehl gebunden. Ein ‘EXEC’ kann aber mehrere ‘DD’ Bausteine enthalten. [\[Ebbe 11\]](#)

2.2. Customer Information Control System

Das Customer Information Control System, kurz CICS, ist ein Applikationsserver für einen IBM-Großrechner. 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 weiteren Vorteil, es unterstützt verschiedene Programmiersprachen. So können Programme innerhalb einer Anwendung in der für ihren Use-Case am besten geeigneten Sprache implementiert werden. Zu den unterstützten Sprachen zählen neben COBOL und IBM Assembler auch Java und Java EE. [\[Rayn 11\]](#)

2.2.1. CICS Transaktion

Ein Businessablauf wird im CICS in einer Transaktion gekapselt. So kann eine Transaktion mehrere Programme unterschiedlicher Programmiersprachen umfassen. Eine Transaktion besitzt ein eindeutiges Kürzel, die TransaktionsID. Über die TransaktionsID kann der Ablauf gestartet werden. Dies kann sowohl per Webanfrage oder per Messaging Queue als auch aus einem anderem Programm heraus oder per Hand geschehen. In der Transaktion werden alle Änderungen die Programme an Ressourcen, wie zum Beispiel einer Datenbank oder Dateien, tätigen protokolliert. So wird im Fehlerfall sichergestellt, dass diese rückgängig gemacht werden können. [\[Rayn 11\]](#)

2.2.2. Voraussetzungen

Im Umfeld der DATEV eG sind die Hard- und Softwarevoraussetzungen um ein CICS beziehungsweise eine CICS Instanz zu erstellen und zu starten vorhanden. Der Fokus dieser Arbeit liegt auf letzterem somit werden nur die dafür notwendigen Voraussetzungen dargestellt. Außerdem liegt der Fokus nur auf Systemen, die vorerst nicht für die produktiven Systeme der DATEV eG vorgesehen sind. Aus diesem Grund werden nur Schritte, die für ein solches Testsystem benötigt werden, dargestellt. Eine weitere Eingrenzung besteht darin, dass nur die Arbeitsschritte, die mit z/OSMF¹ 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 Voraussetzungen aus dem Absatz 2.2.2 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 Falle dieser Arbeit handelt es sich um 17 verschiedene. Diese Dateien benötigt die CICS Instanz um zum Beispiel Systemfehler zu protokollieren. Eine weitere Datei ist dafür zuständig, dass ein Debugger innerhalb der Instanz verwendet werden kann.

2.2.3.2. CSD

In der CICS system definition, kurz CSD, Datei muss jede Ressource, die dem System zur Verfügung stehen soll, definiert werden. Eine CSD Datei kann für mehrere CICS Instanzen verwendet werden. Eine solche allgemeine CSD Datei hat ca. 22.600 Einträge. Ein Eintrag besteht aus einer Gruppe und einer Liste. Die Gruppe ist hierbei die Definition einer Systemressource und muss händisch 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.

¹Beschreibung in Absatz 2.4

2.2.3.3. STC Job

Bei einem Started Task Controll-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.[\[Cass 07\]](#) 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 Dateien². Außerdem werden hier die SIT³ 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 dazugehörigen CSD und 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 [2.2.3](#) 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. [\[Aran 13\]](#)

2.3.1. Queue Manager

Bei einem Queue Manager handelt es sich um die zentrale Ressource eines IBM MQ Systems. So verwaltet er alle anderen IBM MQ Ressourcen. Ausgenommen hiervon ist die Queue-sharing Group, diese ist für diese Arbeit aber nicht von Relevanz. Zur Verwaltung gehören unter anderem die Speichersteuerung der Daten und die Wiederherstellung dieser im Falle eines Fehlers. Desweiteren koordiniert den Zugriff aller Anwendungen auf die Nachrichten in von ihm verwalteten Queues. Um hierbei die Konsistenz sicherzustellen sorgt er für Locking und die notwendige Isolation. [\[Aran 13\]](#)

²Beschreibung in Absatz [2.2.3.1](#)

³CICS system initialization table

2.3.2. Queues

In Queues werden die Nachrichten, die von Programmen gesendet werden gespeichert. Es gibt verschiedene Arten von Queues.

Die für diese Arbeit notwendigen sind zum einen die Local Queue. Dabei handelt es sich um die einzige Queue Art, bei der die Nachrichten physikalisch gespeichert werden. Alle anderen anderen müssen auf eine Local Queue zeigen.

Die andere ist eine spezielle Art von Local Queues, die sogenannten Initiation Queue. Diese dient dem Queue Manager dazu unter bestimmten Bedingungen eine Trigger-Nachricht auf diese 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 das Anwendungen nur starten, wenn wirklich Daten zum Verarbeiten vorhanden sind. [\[Aran 13\]](#)

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 einen 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 ist beispielhaft CICS oder auch WINDOWSNT für Windows unterstützte Plattformen zu nennen. Bei der Anwendung im Fall des CICS Types muss der Name der Transaktion angegeben werden. Für Windows Plattformen der Dateipfad der auszuführenden exe. [\[Aran 13\]](#)

2.4. IBM Cloud Provisioning and Management for z/OS

In diesem Absatz wird zunächst auf die für dieses Kapitel grundlegenden Begriffe eingegangen. Anschließend wird IBM Cloud Provisioning and Management for z/OS, kurz z/OSMF, erläutert. Im Anschluss darauf wird das auf Kommandozeilenbefehle basierende z/OS Provisioning Toolkit, kurz z/OS PT, und dessen Möglichkeiten dargestellt.

2.4.1. Begriffserklärung

Im folgenden werden einige allgemeine Begriffe, die im Umfeld von IBM Cloud Provisioning and Management for z/OS vorkommen, erläutert.

2.4.1.1. 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.

2.4.1.2. Workflow

Ein Workflow ist eine beliebig komplexe eindeutige Aneinanderreihung von sogenannten Steps. Nach der Ausführung dieser wird ein bestimmtes Ziel erreicht, zum Beispiel die erfolgreiche Bereitstellung eines CICS Systems. Die Definition eines Workflows, den dazugehörigen Steps und ihrer Variablen wird in XML umgesetzt. Ein Step beschreibt einen Teilablauf eines Workflows. Innerhalb eines Steps können sowohl interne und externe Scripte als auch JCLs und somit Programme ausgeführt werden. Des weiteren besteht die Möglichkeit REST-Calls zu tätigen. Außerdem 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. Ein weiteres Beispiel ist, es können erforderliche Steps definiert werden, so dass bevor ein Step auf eine Datei zugreift, mittels eines vorherigen Steps geprüft wird ob diese vorhanden ist und wenn nicht diese erzeugt. Es wird ein XML Schema verwendet um sicherzustellen, dass zur Laufzeit keine syntaktischen Fehler vorhanden sind. [Rott 18]

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.

2.4.1.3. Template

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

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

Die nächste Datei ist 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. Dabei ist zu beachten, dass die Datei für die Eingabevariablen und welche Variablen davon verwendet werden, anzugeben ist.

Als letzte Datei ist die Manifest-Datei zu nennen. 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 angegeben. Zusätzlich kann noch eine Beschreibung des Templates hinzugefügt werden.

Somit bildet ein Template einen Rahmen um mehrere Workflows und ermöglicht so schnellere De-/provisionierung. Zudem können die Variablen nur an einer Stelle geändert werden. Außerdem besteht die Möglichkeit, den Variablen zum Zeitpunkt der Provisionierung als Anwendereingabe einen Wert zuzuweisen. Somit ist ein Template flexibler als ein Workflow. [\[IBM 19b\]](#)

2.4.1.4. Instance

Hierbei handelt es sich um das Ergebnis nach der Provisionierung eines Templates. Zum Beispiel eines funktionsfähigen CICS.

2.4.2. IBM Cloud Provisioning and Management for z/OS

Das IBM Cloud Provisioning and Management for z/OS bietet die Möglichkeit mehrere Systeme innerhalb eines z/OS Betriebssystems zu provisionieren, unter anderem Laufzeitumgebungen wie CICS. Jedoch nicht die Bereitstellung eines kompletten z/OS Betriebssystems. Für diese Aufgaben stehen zwei Schnittstellen zur Verfügung. Zum einem z/OS Provisioning Toolkit, im Weiteren z/OSPT genannt, und zum anderen z/OS Management Facility, im Weiteren z/OSMF genannt. [\[Keit 16\]](#)

2.4.2.1. z/OS Provisioning Toolkit

z/OSPT bietet ein Kommandozeileninterface für die Bereitstellung und das Verwalten von Laufzeitumgebungen. In Abbildung 2.1 werden die möglichen Kommandozeilenbefehle mittels des Befehls ‘zospt -h’ in einem Kommandofenster angezeigt. Mit z/OSPT werden noch zwei weitere Begriffe eingeführt.

Zum einen sogenannte ‘images’. Dabei handelt es sich grundsätzlich um ein Template, jedoch kann dieses Template über eine weitere Inputfile verändert werden. Dadurch kann ein Template mit kleineren Änderungen provisioniert werden, ohne dass ein neues Template erzeugt werden muss. Dies erhöht die Flexibilität weiter.

Des anderen die sogenannten ‘container’. Dabei handelt es sich eins zu eins um eine Instance. [\[IBM 19a\]](#)

```

$ 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

2.4.2.2. z/OS Management Facility

Der Hauptaugenmerk dieser Arbeit liegt jedoch bei z/OSMF. Da dieses die Verwaltung von Workflows und Templates über eine browserbasierende Schnittstelle ermöglicht. Durch diese Oberfläche, in Abbildung ?? dargestellt, ist es einfacher zu bedienen und somit wird der Einstieg in die Provisionierung erleichtert.

//hier zosmf welcomepage screenshot

Wie auf der rechten Seite der Abbildung ?? zu sehen ist, bietet z/OSMF viele Funktionen an. Für diese Arbeit besitzt nur der Menüpunkt ‘Cloud Provisioning’ Relevanz. Unter diesem Punkt sind die Funktionalitäten für die automatisierte Bereitstellung von Templates zu finden. [Rott 18]

Dabei handelt es sich zunächst um das ‘Resource Management’. Darunter werden sogenannte ‘Domains’ und die dazugehörigen ‘Tenants’ verwaltet. Unter einer ‘Domain’ ist ein System, das Systemressourcen in Ressourcenpools gliedert, zu verstehen. ‘Tenants’ sind die dazugehörigen Rechtegruppen, die dem Anwender den Zugriff und die Nutzung von zugeordneten Templates ermöglicht. Einem Template muss sowohl eine ‘Domain’ als auch ein ‘Tenant’ zugewiesen werden. [Rott 18]

Zur Verwaltung der Templates und Instances kommen die ‘Software Services’ zum Einsatz. Dort können neue Templates über Manifest Datei hinzugefügt werden. Dann muss, wie oben beschrieben, eine ‘Domain’ und ein ‘Tenant’ zugewiesen 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 kann damit das Template und die in der Aktion-Definitions-Datei definierten Aktionen getestet werden. [\[Rott 18\]](#)

Kapitel 3.

Vorgehensweise

Zu Beginn dieser Arbeit war eine Einarbeitung nicht nur in das verwendete Toolkit notwendig. Sondern es musste sich viel mit den verschiedenen Systemen, also CICS, Db2 und MQ, im Hinblick auf administrative Aufgaben auseinandergesetzt werden. Nachdem eine Beispielanwendung gefunden wurde, folgte die Analyse des Ist-Zustandes inklusive der Beschreibung dieser Anwendung. Während der Analyse wurde der momentane Bereitstellungsprozess untersucht.

(evtl. Workshop erwähnen)

Die Installation des Toolkits geschah bereits vor dem Beginn dieser Arbeit. Somit ist es möglich auf dem Testplex, einer Systemumgebung für Test von neuen Betriebssystemversionen oder ähnlichem, zu beginnen. Dieser wird hauptsächlich von Administratoren genutzt. Zusätzlich ist dieser 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. 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 Entwicklungsumgebung. Letzteres ist die Testumgebung für alle Mainframe Entwickler. Hier werden vor allem neue Programmversionen entwickelt und damit kleinere Tests durchgeführt. Außerdem sind in der Entwicklungsumgebung alle Anwendungsdaten, die für diese Tests notwendig sind, vorhanden. Somit kann die Integration der Beispielanwendung in das provisionierte CICS stattfinden.

Zuletzt wird eine Diskussionsrunde stattfinden. Hierbei werden Kollegen aus allen beteiligten Gruppen teilnehmen. Dazu zählen CICS-Administratoren, Db2-Administratoren, MQ-Administratoren, Entwickler der Beispielanwendung und Architekten. Zunächst wird das

Ergebnis dieser Arbeit vorgestellt. An Hand dessen wird diskutiert, ob und wenn ja wie das ‘IBM Cloud and Management for z/OS Toolkit‘ verwendet werden soll.

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 Batch¹-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.

¹Stapelverarbeitung

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 DMVIN²-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 verhindern, 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.

²DatevMakroVerarbeitungsinformation

Mittels dieser drei Möglichkeiten werden insgesamt monatlich circa 30 Millionen Datensätze bereitgestellt und weiterverarbeitet. Diese Datensätze stehen innerhalb der durch das DMVINP-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 ARUBA³-Db2-Datenbank. Dort werden die Berechnungsdaten der letzten 36 Monate aufbewahrt. Dabei handelt es sich 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 Rechnungsstellungsrhythmen (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 4.1.2 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 Rechnungsnettoeträge geprüft, ob diese über einem bestimmten Rechnungslimitbetrag liegen. Falls dies nicht der Fall ist, werden die Berechnungssätze als BUL⁴ gekennzeichnet und in die folgende Rechnungsperiode vorgetragen. Schließlich wird noch die Umsatzsteuer ermittelt.

³Abrechnungs- und Umsatz-Basis

⁴Berater 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 Queueamen 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

Mit vielen anderen Abteilungen sprechen

Viel auf 'Zuruf' und Besprechungen

Genauere Infos noch von den CICSAdmins nachfragen

Kapitel 5.

Realisierung

In diesem Kapitel wird beschrieben, wie die Aufgabe dieser Arbeit gelöst wurde. Dazu wird nach der im Kapitel 3 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.

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 mitgeliefertes 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 eine Domain und einen Tenant zuzuweisen. Dies hatte zur Folge das der Befehl ‘zospt build’

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 Prefix bei Authentificatio- nanfragen 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	Homeverzeichnis des Unix System Services
DFH_CICS_HLQ	High-level qualifier von dem CICS Installationsort.

Tabelle 5.1.: Zu verändernde Variablen im minimalen CICS Template

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 5.1 genannten Variablen angepasst werden. Die Kurzbeschreibungen und die Beschreibungen aller Variablen, die im Standard Template vorhanden sind, ist in [IBM 19b] 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 Dateien³ 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 Namenskonventionen gespeichert werden. Somit ist sichergestellt, dass durch die neu provisionierten Instanzen die Alten nicht beeinflusst werden. Außerdem kann jeder Anwender 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,

³Beschreibung in Absatz 2.2.3.1

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 erklären..... bzw job im grundlagen teil erklären

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 2.4.1.2 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 Abildung 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 5.2.3 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 5.1.1.2 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 Rechtsvorschriften 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 User⁴ 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. Als nächstes musste ein Parameter bei der Erstellung der CICS spezifischen Dateien hinzugefügt werden. Es handelt sich um den Messageclass 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

⁴User ID mit zunächst keinen Berechtigungen

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 Preisabhängigkeiten. 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 [A.1](#) zu finden. Dabei handelt es sich um ungetesteten Code. Nachdem dieser Code getestet 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 [4.1.2](#) beschrieben, sehr viele gleichartige Queues benötigt werden, wurde für die Erstellung dieser von den MQ Administratorenteam ein Rexx Skript angefertigt. Dies

geschah unabhängig 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 Queue-Manager die Queues angelegt werden und den ersten Qualifier des Queue-Namens. Für den restlichen Queue-Namen existiert auch eine Variable, in dieser werden die Namen als Komma-separierte Liste angegeben und ausgelesen. Anhand dieser Namen wird dann die maximale Queue-Tiefe 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 entwickelt, 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 Queue-Tiefe 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 5.1.1.3 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.

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 Provisioning’. Anschließend öffnet 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. Da 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 Workflow-datei durchführt, 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 5.2.2 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 5.1 als Beispiel, von zOSMF zu nennen. So wird bei dem Hinzufügen und Aktualisieren

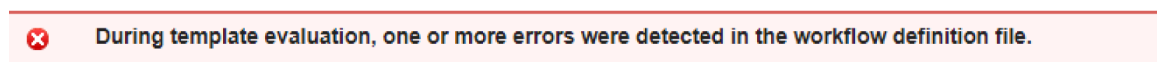


Abbildung 5.1.: Beispiel einer Fehlermeldung von zOSMF

eines Templates in zOSMF das Template und damit alle davon benötigten Dateien auf Syntaxfehler geprüft. Die in Abbildung 5.1 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 Rxxvariablen 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_QUEUEENAMES‘ 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.

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. 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. Eine Herangehensweise an dieses Problem wird im Ausblick, Kapitel 6, dargestellt.

5.5. Interviews

Kapitel 6.

Ausblick

Bezug auf den Anfang

Integrieren in eine Buildpipeline mit im allgemeinen zospt

Problem des zu spezifischen Templates

Möglichkeit (NUR MÖGLICHKEIT) Bereitstellen der CICSe auch in Produktion

Kapitel 7.

Zusammenfassung

Anhang A.

Anhang

A.1. Produktstammdaten data definition language

```
1  CREATE TABLE PSSSCHEMA.TPAUSPSSBASELBART
2      (PARTITIONID INTEGER
3
4      NOT NULL
5  WITH DEFAULT 1
6      ,PID INTEGER
7
8      NOT NULL
9      ,AUSPRAEGUNG SMALLINT
10
11     NOT NULL
12     ,GUELTIGAB DATE
13
14     NOT NULL
15     ,LFDNR INTEGER
16
17     NOT NULL
18     WITH DEFAULT
19     ,GUELTIGBIS DATE
20
21     NOT NULL
22     WITH DEFAULT "9999-12-31 "
23     ,PSSID INTEGER
24     WITH DEFAULT NULL
25     ,ZUSATZID CHARACTER(4) FOR SBCS DATA
26     WITH DEFAULT NULL
27     ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
28     WITH DEFAULT NULL
29     ,MWSTANTEILFREI DECIMAL(5, 2)
30     WITH DEFAULT NULL
31     ,MWSTANTEILREDUZIERT DECIMAL(5, 2)
32     WITH DEFAULT NULL
33     ,MWSTANTEILVOLL DECIMAL(5, 2)
34     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
365                                     NOT NULL
366     WITH DEFAULT 1
367         ,PID INTEGER
368
369                                     NOT NULL
370         ,ARTNR INTEGER
371
372                                     NOT NULL
373     WITH DEFAULT "1966-02-14"
374         ,OPDATBEN CHARACTER(1) FOR SBCS DATA
375
376                                     NOT NULL
377     WITH DEFAULT "J"
378         ,AUSDIENSTREL CHARACTER(1) FOR SBCS DATA
379
380                                     NOT NULL
381     WITH DEFAULT "N"
382         ,VERTRIEBSREL CHARACTER(1) FOR SBCS DATA
383
384                                     NOT NULL
385     WITH DEFAULT NULL
386         ,KOMMASTELLEN INTEGER
387     WITH DEFAULT NULL
388         ,ERTRNR INTEGER
389
390                                     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.TPAUSPSSBART
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.TPAUSPSSBART
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;

Abbildungsverzeichnis

1.1. Annual amount of unplanned server downtime worldwide in 2019, by hardware platform	2
2.1. z/OSPT mögliche Kommandozeilenbefehle	12
5.1. Beispiel einer Fehlermeldung von zOSMF	32

Tabellenverzeichnis

5.1. Zu verändernde Variablen im minimalen CICS Template	24
--	----

Quellcodeverzeichnis

listings/ddl.txt	39
----------------------------	----

Literaturverzeichnis

- [Also 93] S. Alsop. “IBM still has the brains to be a player in client/server platforms”. *InfoWorld*, Vol. 15, No. 10, p. 4, 1993.
- [Aran 13] C. Aranha. *IBM WebSphere MQ V7.1 and V7.5 features and enhancements*. IBM redbooks, IBM Corp. International Technical Support Organization, Poughkeepsie, NY, 1st ed. Ed., 2013.
- [Cass 07] P. Cassier. *System programmer’s guide to Workload manager*. IBM redbooks, IBM International Technical Support Organization, United States?, 4th ed. Ed., 2007.
- [Ceru 03] P. E. Ceruzzi. *A history of modern computing*. *History of computing*, MIT Press, Cambridge, Mass., 2. ed. Ed., 2003.
- [DATE 17] DATEV eG. “Geschichte der Datev”. 2017.
- [Ebbe 11] M. Ebberts, J. Kettner, W. O’Brien, and B. Ogden. *Introduction to the new mainframe: Z/OS basics*. IBM redbooks, IBM Corporation International Technical Support Organization, Poughkeepsie, NY, third edition , (march 2011) Ed., 2011.
- [IBM 14] IBM. “Access method control block (ACB)”. 2014.
- [IBM 19a] IBM. “CICS Application Server Software for IBM Z”. 2019.
- [IBM 19b] IBM. “Using IBM z/OS Provisioning Toolkit”. 2019.
- [Keit 16] Keith Winnard, Gary Puchkoff, Hiren Shah. *IBM Cloud Provisioning and Management for z/OS: An Introduction*. Redbooks, IBM International Technical Support Organization, Poughkeepsie, N.Y., 2016.
- [Kuhn 19] J. B. Kühnapfel. *Nutzwertanalysen in Marketing und Vertrieb. essentials*, Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2. auflage 2019 Ed., 2019.
- [Rayn 11] C. Rayns. *CICS transaction server from start to finish*. Redbooks, IBM International Technical Support Organization, Poughkeepsie, N.Y., 2011.
- [Roge 11] P. Rogers. *ABCs of z/OS system programming: Volume 4*. IBM redbooks, IBM International Technical Support Organization, Poughkeepsie, N.Y.?, 2011.

- [Rott 18] R. J. T. Rotthove. *IBM z/OS Management Facility V2R3. Redbooks*, IBM Redbooks, [Place of publication not identified], 2018.
- [Sull 16] D. Sullivan. “Google now handles at least 2 trillion searches per year - Search Engine Land”. 2016.
- [TIOB 19] TIOBE Software BV. “TIOBE Index | TIOBE - The Software Quality Company”. 25.11.2019.