IBM Cúram Social Program Management
Version 6.0.5

Handbuch Cúram Universal Access

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen in „Bemerkungen" auf Seite 103 gelesen werden.

Überarbeitung: März 2014

Diese Ausgabe bezieht sich auf IBM Cúram Social Program Management v6.0.5 und alle nachfolgenden Releases, sofern nicht anderweitig in neuen Ausgaben angegeben.

Licensed Materials - Property of IBM.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Universal Access anpassen

In diesem Handbuch erfahren Sie, wie Sie die folgenden Universal Access-Komponenten anpassen können: CitizenWorkspace, CitizenWorkspaceAdmin und WorkspaceServices. Die wichtigsten anpassbaren Funktionsbereiche sind Triage, Screening, Anliegen, Sicherheit, Bürgerkonto und Lebensereignisse.

Einführung

Zweck

Der Zweck dieses Handbuchs besteht darin, die Optionen für die Anpassung von IBM Cúram Universal Access-Komponenten sowie die entsprechende Vorgehensweise zu beschreiben. Eine Anpassung unterscheidet sich dahingehend von einer Konfiguration, dass Entwickler bei einer Anpassung in der Lage sind, Quellcode so zu ändern, zu erweitern oder zu ersetzen, dass er bestimmte Kundenanforderungen erfüllt. Die in diesem Zusammenhang relevanten Komponenten, aus denen Universal Access besteht, sind 'CitizenWorkspace', 'CitizenWorkspaceAdmin' und 'WorkspaceServices'. Bei den in diesem Dokument erläuterten anpassbaren Funktionsbereichen geht es hauptsächlich um Triage, Screening, Anliegen, Sicherheit, Bürgerkonto und Lebensereignisse. Lesen Sie hierzu die Informationen im *Handbuch zu Cúram Universal Access*, um sich mit diesen Konzepten besser vertraut zu machen.

Das Produkt IBM Cúram Universal Access wird durch die vorstehend aufgeführte Gruppe von Komponenten implementiert. Diese Komponenten werden zusammen als Citizen-Arbeitsbereich oder als Bürgerarbeitsbereich bezeichnet.

Das Akronym UA (Universal Access) wird nicht im Zusammenhang mit Cascading Style Sheets verwendet.

Zielgruppe

Dieses Handbuch richtet sich an Entwickler, die für die Anpassung von Universal Access-Komponenten verantwortlich sind.

Geltungsbereich

Das vorliegende Handbuch behandelt die Anpassung der Komponenten 'CitizenWorkspace', 'CitizenWorkspaceAdmin' und 'WorkspaceServices'. Kunden, die eine Lizenz für Cúram Enterprise Framework, aber nicht für Universal Access besitzen, brauchen sich nur mit Arbeitsbereichsservices im Zusammenhang mit Anpassungspunkten zu beschäftigen.

Anmerkung: Das vorliegende Handbuch behandelt nicht die Konfigurationsoptionen, die für die Komponente Universal Access verfügbar sind. Mithilfe einer Konfiguration können Administratoren festlegen, welche Informationen auf Anwendungsseiten angezeigt werden. Weitere Informationen zur Konfiguration enthält das Handbuch zur Cúram Universal Access-Konfiguration (`Cúram Universal Access Configuration Guide`).

Informationen

Vor der Lektüre dieses Handbuchs sollten Sie mit dem Inhalt der nachstehenden Entwicklerhandbücher vertraut sein. In diesen Entwicklerhandbüchern werden die

Grundlagen dazu erläutert, wie das Verhalten der Komponente Universal Access konfiguriert wird und wie Anpassungen in der Produktgruppe im Allgemeinen und für Ereignisse und Workflows im Besonderen vorgenommen werden können.

- Engine für Datenzuordnung verwenden (`Using the Data Mapping Engine`)
- Handbuch zu Cúram Universal Access (`Cúram Universal Access Guide`)
- Handbuch zur Cúram-Entwicklungskonformität (`Cúram Development Compliancy Guide`)
- Handbuch für Cúram-Serverentwickler (`Cúram Server Developer's Guide`)
- `Persistence Cookbook`
- Handbuch zum Workflow-Management-System (`Cúram Workflow Management System Guide`)

Kapitel in diesem Handbuch

Antragsbanner, Antragsmenüs & Navigation konfigurieren
In diesem Kapitel werden die Konfigurationsoptionen für Banner, Menüs und Navigation beschrieben.

Blackbox-Entwicklungsphilosophie
In diesem Kapitel wird das Engagement von Universal Access hinsichtlich der Blackbox-Entwicklungsphilosophie kurz erläutert.

Triage anpassen
In diesem Kapitel werden die vorhandenen Anpassungspunkte im Zusammenhang mit dem Triage-Prozess erläutert.

Universal Access-Screening anpassen
In diesem Kapitel werden die Anpassungspunkte im Zusammenhang mit dem Screening-Prozess erläutert.

Verarbeitung von Anliegen in Anträgen anpassen
In diesem Kapitel werden die vorhandenen Anpassungspunkte im Zusammenhang mit dem Anliegenprozess vor der Übermittlung erläutert.

Bearbeitung übermittelter Anträge anpassen
In diesem Kapitel werden die Anpassungspunkte im Zusammenhang mit dem Anliegenprozess nach der Übermittlung erläutert.

Bürgerkonto anpassen
In diesem Kapitel werden die vorhandenen Anpassungspunkte im Zusammenhang mit dem Bürgerkonto erläutert.

Lebensereignisse anpassen
In diesem Kapitel werden die Anpassungspunkte im Zusammenhang mit Lebensereignissen erläutert.

Universal Access-Web-Services
In diesem Kapitel werden die Web-Services von Universal Access beschrieben, und es wird erläutert, wie Peer-Code entwickelt werden kann, um mit diesen Web-Services zu kommunizieren.

Motivationen
In diesem Kapitel werden die Motivationen in Universal Access beschrieben, und es wird erläutert, wie eine Motivation implementiert wird.

Vollständig anpassbare Universal Access-Artefakte
In diesem Kapitel werden die bereitgestellten Artefakte beschrieben, die vollständig anpassbar sind.

Universal Access-Artefakte mit eingeschränkten Möglichkeiten für die Anpassung In diesem Kapitel werden diejenigen bereitgestellten Artefakte angegeben, deren Verwendung bestimmten Einschränkungen unterliegt.

Blackbox-Entwicklungsphilosophie

Einführung

Universal Access ist ein Blackbox-Produkt. Dies bedeutet, dass die Entwicklung dieses Produkts von Anfang an auf extrem hohe Flexibilität ausgerichtet wurde, mit der Möglichkeit, viele Aspekte der Produktfunktionalität während der Laufzeit einfach durch entsprechende Konfigurationen zu ändern. Viele weitere Aspekte können mithilfe der APIs von Universal Access geändert werden. Wenn es nach der Lektüre dieses Handbuchs immer noch gewünschte Aktionen gibt, die Sie nicht ausführen können, haben Sie die Möglichkeit, eine entsprechende Funktion in einem Service-Pack oder in einem weiteren zukünftigen Release anzufordern. Wenn Sie diesen Weg wählen, wird die bereitgestellte Funktion mitsamt den zugehörigen Tests und der entsprechenden Qualitätssicherung in das Produkt integriert.

Zeit- und Kosteneinsparungen ermöglichen

Ein als Blackbox entwickeltes Produkt kann Ihnen dabei helfen, Zeit und Kosten zu sparen. IBM hat es sich zur Aufgabe gemacht, auf Anforderungen nach Erweiterungen zeitnah zu reagieren. Dadurch wird sichergestellt, dass Sie keine Zeit und Kosten in die Entwicklung von Erweiterungen investieren müssen, einschließlich der entsprechenden Unterstützung und Kosten, die damit einhergehen. Darüber hinaus ist IBM dahingehend engagiert, sinnvolle Erweiterungen und neue Konfigurationen für das Produkt zu entwickeln, wo bisher keine vorhanden sind.

Vereinfachte Upgrades

Universal Access ist eine strategische Plattform zur Implementierung der Dienstleistungen von Sozialdiensten und -behörden für deren Kunden.

- Universal Access ist eine Plattform: Sie bietet eine Gruppe von Anwendungsprogrammierschnittstellen (APIs) und Erweiterungspunkten, mit deren Hilfe eine Lösung erstellt werden kann, die den individuellen Anforderungen einzelner Kunden entspricht.
- Universal Access ist ein strategisches Produkt: Bei der Entwicklung und Erstellung dieses Produkts wurden von Anfang an Überlegungen zu zukünftigen Upgrades berücksichtigt. Dadurch soll erreicht werden, dass Upgrades einfach durchzuführen sind und jeweils eine Reihe neuer, abwärtskompatibler Funktionen mit sich bringen.

Der zweite Punkt ist ein wichtiger Aspekt der Universal Access-Entwicklungsphilosophie: Durch die Verwendung und Erweiterung von Universal Access in der empfohlenen Art und Weise können Sie neue Versionen mit minimalem Aufwand übernehmen und so alle neuen Funktionen, die im Rahmen von Upgrades geboten werden, für Ihre Arbeit nutzen. Wenn Kunden von den in diesem Dokument empfohlenen Richtlinien abweichen, besteht ein erhöhtes Risiko, dass bei einem Upgrade Schwierigkeiten auftreten.

Möglichkeit zur Konfiguration und Anpassung

Im Einklang mit unserem Engagement für die Blackbox-Entwicklungsphilosophie stehen bereits jetzt zahlreiche Anpassungs- und Konfigurationsoptionen zur Verfügung. Die Universal Access-Plattform wurde so erstellt, dass sie so viele Konfigura-

tionspunkte wie möglich ohne Vorbereitungs- oder Anpassungsaufwand abdeckt. Die entsprechenden Optionen werden im vorliegenden Dokument erläutert.

Antragsbanner, Antragsmenüs & Navigation konfigurieren

Einführung

Es gibt zwei wichtige Aspekte bei der Konfiguration des neuen externen Antrags:

- Das Antragsbanner, einschließlich des übergeordneten Megamenüs und anderer Menüs.
- Die Navigationsleiste.

Derzeit gibt es ein Antragsbanner in der Konfigurationsdatei der internen Anwendung (mit der Dateierweiterung .app), sodass dieser Inhalt zwecks Unterstützung der externen Anträge wiederverwendet und erweitert wird.

In ähnlicher Weise werden die vorhandenen Konfigurationsdateien für die Navigation (mit der Erweiterung .nav) verwendet, um die neue externe Navigationsleiste zu unterstützen.

Antragskonfiguration

Das vollständige Schema für die vorhandenen .app-Dateien befindet sich im Ordner 'JDECommons/lib/schema', insbesondere in der Datei 'application-view.xsd'.

Im Folgenden wird der zusätzliche Inhalt beschrieben, der hinzugefügt wird, und es wird dargelegt, an welcher Stelle vorhandener Inhalt in einem anderen Kontext verwendet wird.

Relevante Attribute im Element *application* für Anträge:

Tabelle 1. Relevante Attribute im Element application

Attribut	Beschreibung/Verwendung	Neu/Vorhanden
id	Vorhandenes Attribut, das die eindeutige ID des Antrags identifiziert und mit dem Namen der Datei übereinstimmen muss. Bei internen Anträgen besteht hier ein Link zur Codetabelle APPLICATION_CODE und der Startseite des Benutzers. Bei externen Anträgen ist dies nicht der Fall.	Vorhanden
title	Ein Verweis auf den Inhalt in der Datei '.properties'. Dieser Inhalt wird nicht im Antragsbanner angezeigt, sondern in den Administrationsfenstern zur Identifizierung des internen Antrags verwendet.	Vorhanden
subtitle	Ein Verweis auf den Inhalt in der Datei '.properties'. Dieser Inhalt wird nicht im Antragsbanner angezeigt, sondern in den Administrationsfenstern zur Identifizierung des internen Antrags verwendet.	Vorhanden
mode	Ist dieses Attribut nicht gesetzt, wird davon ausgegangen, dass die Datei '.app' für einen internen Antrag verwendet wird. Ist dieses Attribut auf 'external' gesetzt (dies ist momentan der einzige unterstützte Wert), bedeutet dies, dass die Datei '.app' einen externen Antrag definiert. Dieses Attribut wird verwendet, um Inhalt und Validierung für beide Antragstypen unterschiedlich zu bearbeiten.	Neu

Alle anderen Attribute werden nicht unterstützt und für .app-Dateien mit mode="external" ignoriert.

Bei den folgenden Elementen handelt es sich um neue Elemente, die als direkte untergeordnete Elemente des Elements *application* unterstützt werden. Sämtliche Elemente sind optional.

- *landing-page*

  Das Symbol und der Inhalt, die in der linken Ecke des Antragsbanners angezeigt werden, einschließlich des Hyperlinks zur sogenannten Landing-Page (Zielseite). Wird ein Antrag erstmals geladen, wird diese Seite geöffnet, sofern sie definiert ist. Dies unterscheidet sich von einem internen Antrag, bei dem der Wert für APPLICATION_CODE des Benutzers verwendet wird. Hinweis: Wird kein Element 'page-id' oder 'landing-page' angegeben, dann wird der erste Eintrag in der Navigation als Landing-Page (Zielseite) verwendet. Es ist möglich, kein Element 'landing-page' zu definieren. In diesem Fall zeigt der Renderer nichts an.

Tabelle 2. Relevante Attribute im Element 'landing-page'

Attribut	Beschreibung	Erforderlich
title	Ein Verweis auf den unter dem Symbol angezeigten Text.	Ja
page-id	Die Seite, die geöffnet werden soll, wenn auf das Symbol/den Text geklickt wird.	Ja
icon	Ein Verweis auf das Bildsymbol, das angezeigt werden soll.	Nein

- *banner-menu*

  Für das Element 'banner-menu' werden im Antragsbanner drei Typen unterstützt:

  – mega - Ein übergeordnetes Mega-Hauptmenü, das als erstes Menü angezeigt wird.

  – person - Das Personenmenü, das normalerweise den Benutzernamen/die Willkommensnachricht und die Optionen für Abmeldung anzeigt.

  – help - Ein Menü mit Links zu Hilfeinformationen/Kontakten. Die Darstellung dieses Menüs ähnelt dem Mega-Menü.

  Es ist möglich, keine Elemente vom Typ 'banner-menu' zu definieren. In diesem Fall zeigt der Renderer nichts an. Darüber hinaus muss ein Bannermenü keine Menüeinträge enthalten. Auch in diesem Fall zeigt der Renderer nichts an. Diese Elemente sind also optional.

Tabelle 3. Relevante Attribute im Element 'banner-menu'

Attribut	Beschreibung	Erforderlich
type	Derzeit werden folgende Werte unterstützt: 'mega', 'help' und 'person'. Darüber hinaus kann jeder Typ jeweils nur einmal definiert werden.	Ja
title	Ein Verweis auf den Titeltext, der angezeigt werden soll.	Ja
page-id	Ein Verweis auf die Seite, die geöffnet werden soll. Dieses Attribut ist optional und wird nur für das Personenmenü in der ersten Version unterstützt.	Nein

Ein Element *banner-menu* kann 0 bis n untergeordnete Elemente vom Typ *menu-item* mit folgenden Attributen enthalten:

Tabelle 4. Relevante Attribute in Element 'menu-item'

Attribut	Beschreibung	Erforderlich
id	Eine eindeutige ID für den Menüeintrag. Diese ID muss innerhalb der Datei eindeutig sein.	Ja
icon	Ein Verweis auf das Symbol, das angezeigt werden soll.	Nein
title	Ein Verweis auf den Titeltext.	Ja

Tabelle 4. Relevante Attribute in Element 'menu-item' (Forts.)

Attribut	Beschreibung	Erforderlich
text	Ein Verweis auf einen längeren Beschreibungstext.	Nein
page-id	Ein Verweis auf die Seite, die geöffnet werden soll.	Ja

Ein Element *menu-item* kann 0 bis n untergeordnete Elemente vom Typ *param* enthalten. Bei den Elementen vom Typ 'param' handelt es sich um fest codierte Werte, die als Parameter an den Link übergeben werden, der dem Element 'menu-item' zugeordnet ist. Elemente vom Typ 'param' haben folgende Attribute:

Tabelle 5. Relevante Attribute im Element 'param'

Attribut	Beschreibung	Erforderlich
name	Der Name des Parameters, der mit dem Link übergeben werden soll.	Ja
value	Ein Verweis auf den Wert des Parameters. Dieser Wert kann lokalisiert werden. Wenn keine Lokalisierung erfolgt, wird der Wert unverändert übergeben.	Ja

Ein Parameter wird verwendet, um die 'Motivation' des Links festzulegen. In den meisten Fällen lautet der Name daher 'Motivation' mit einem Wert. Der Wert kann lokalisiert werden. Wenn er jedoch nicht in der Datei '.properties' vorhanden ist, wird der in der XML-Datei angegebene Wert gesendet.

- *navigation*

  Ein Verweis auf die Navigationsdatei (.nav), die die Liste der Einträge enthält, die in der Navigationsleiste für den Antrag angezeigt werden sollen. Navigationsleisten werden normalerweise auf Registerkartenebene definiert. In diesem Fall gilt sie jedoch für den gesamten Antrag.

Tabelle 6. Relevante Attribute im Element 'navigation'

Attribut	Beschreibung	Erforderlich
id	Ein Verweis auf die ID der Datei '.nav'; die ID ist gleichzeitig der Name der Datei, beispielsweise '<id>.nav'.	Ja
width	Ein Verweis auf den Wert, der als Breite der Navigationsleiste in Pixel verwendet werden soll. Dies ermöglicht eine individuelle Konfiguration der Lokalisierung pro Antrag.	Nein

Navigationskonfiguration

Das vollständige Schema für die vorhandenen .app-Dateien befindet sich im Ordner 'JDECommons/lib/schema', insbesondere in der Datei 'application-view.xsd'.

Im Folgenden wird der zusätzliche Inhalt beschrieben, der hinzugefügt wird, und es wird dargelegt, an welcher Stelle vorhandener Inhalt in einem anderen Kontext verwendet wird.

Relevante Attribute im Element *application*:

Tabelle 7. Relevante Attribute im Element application

Attribut	Beschreibung/Verwendung	Neu/Vorhanden
id	Die ID für die Navigationsdatei; diese ID muss mit dem Namen der Datei übereinstimmen.	Vorhanden

Tabelle 7. Relevante Attribute im Element application (Forts.)

Attribut	Beschreibung/Verwendung	Neu/Vorhanden
loader-registry	Eine Liste der Ladeprogramme, die verwendet werden können, um die Anzeige von Inhalt dynamisch zu steuern.	Vorhanden
nodes	Die Liste der Navigationselemente.	Vorhanden

Das Element *nodes* unterstützt zwei untergeordnete Elemente:

- *navigation-group*

  Dieses untergeordnete Element gilt nicht für die Navigation externer Anträge.
- *navigation-page*

  Der Navigation können 1 bis n Elemente vom Typ *navigation-page* hinzugefügt werden. Jedes Element stellt einen Link in der Navigationsleiste dar. Folgende Attribute sind gültig:

Tabelle 8. Relevante Attribute im Element 'navigation-page'

Attribut	Beschreibung/Verwendung	Neu/Vorhanden
id	Die eindeutige ID des Eintrags, die verwendet wird, um Konflikte während der Ergänzung zu vermeiden.	Vorhanden
title	Ein Verweis auf den Titel des Navigationselements.	Vorhanden
description	Ein Verweis auf eine Administratorbeschreibung für das Element.	Vorhanden
visible	Ein boolescher Wert, der angibt, ob das Element standardmäßig sichtbar ist oder nicht.	Vorhanden
dynamic	Ein boolescher Wert, der angibt, ob die Sichtbarkeit des Elements von einem Ladeprogramm gesteuert werden kann.	Vorhanden
page-id	Ein Verweis auf die Seite, die geöffnet werden soll, wenn das Element angeklickt wird.	Vorhanden
icon	Ein Verweis auf das Symbol, das für den Eintrag angezeigt werden soll.	Neu

Ein neues untergeordnetes Element namens *highlight* wird unter dem Element *navigation-page* nur im Kontext eines externen Antrags unterstützt. Dieses Element enthält lediglich ein Attribut; es können jedoch 0 bis n Einträge definiert werden:

Tabelle 9. Relevante Attribute im Element 'highlight'

Attribut	Beschreibung
page-id	Die ID einer UIM-Seite, die beim Anzeigen im Inhaltsbereich dazu führt, dass das betreffende Navigationselement hervorgehoben wird.

Schutz von Universal Access

Einführung

Das vorliegende Kapitel dient dem folgenden Zweck:

- Lesern einen Überblick über das Sicherheitsmodell von Universal Access zu verschaffen
- Entwicklern die Vorgehensweise beim sicheren Anpassen von Universal Access zu vermitteln

Hintergrund

Universal Access wurde konzipiert, um Bürgern über das Internet Zugriff auf ihre höchst sensiblen persönlichen Daten zu ermöglichen. Daher muss Sicherheit oberste Priorität haben, wenn Anpassungen für das Bürgerkonto entwickelt werden. Alle in Universal Access erstellten Projekte müssen schwerpunktmäßig auf die Bereitstellung dieser Sicherheit ausgerichtet sein. Dafür ist es erforderlich, dass das Projektteam die Sicherheit von Anfang an in seine Überlegungen einschließt und diese nicht nur am Ende einfach einer Prüfung unterzieht. Es wird empfohlen, bei allen Projekten mindestens die nachstehenden Schritte auszuführen, um die Sicherheit der übermittelten Daten zu gewährleisten. Im verbleibenden Teil dieses Kapitels wird das Universal Access-Sicherheitsmodell erläutert. Darüber hinaus erhält der Leser Informationen dazu, wie Anpassungen hinzugefügt werden können, die nicht gegen, sondern für dieses Sicherheitsmodell arbeiten.

- Stellen Sie sicher, dass das Projektteam mit den Grundsätzen der Entwicklung sicherer Anwendungen sowie mit häufig auftretenden Sicherheitsproblemen wie beispielsweise den OWASP Top Ten vertraut ist.
- Entwickeln Sie ein Sicherheitsrisikomodell (Threat Model) und wenden Sie dieses an.
- Beauftragen Sie Sicherheitsexperten mit dem Testen aller relevanten Aspekte - von den Anforderungen bis hin zur fertigen Implementierung.
- Planen Sie, wie die Anwendung in öffentlich zugänglichen Bereichen wie beispielsweise Bibliotheken und Kiosks verwendet werden soll.

Universal Access-Sicherheitsmodell

Universal Access verfügt über eine Reihe unterschiedlicher Kontotypen, um sowohl anonyme als auch registrierte Benutzer zu unterstützen, die diese Anwendung nutzen. Bei fortlaufender Verwendung von Universal Access kommen die Benutzer mit einer Reihe dieser unterschiedlichen Kontotypen in Berührung. In diesem Abschnitt werden die verschiedenen Kontotypen vorgestellt und deren jeweiliger Zweck erläutert. Die Benutzertypen können wie folgt zusammengefasst werden:

- Öffentliches Bürgerkonto
- Anonyme Konten
- Registrierte Konten
- Verknüpfte Konten

Öffentliches Bürgerkonto

Wenn der Benutzer die Titelseite von Universal Access anzeigt, wird er automatisch unter dem Konto `publiccitizen` angemeldet. Dieses Konto verfügt lediglich über Zugriff auf die Startseite sowie auf Seiten zum Eingeben oder Zurücksetzen von Kennwörtern.

Anonyme Konten

Wenn der Benutzer auf einen Link klickt, um eine Triage, ein Screening oder eine Anliegenaufnahme durchzuführen, wird er automatisch als `publiccitizen` abgemeldet und anschließend unter einem anonymen Konto mit einem willkürlich generierten Benutzernamen erneut angemeldet. Dieser Benutzername kann für die Dauer einer Sitzung verwendet werden, in der der Benutzer eine Triage, ein Screening und/oder eine Anliegenaufnahme durchführen kann. Es gibt gute Sicherheitsgründe dafür, jede einzelne Sitzung einem anderen generierten Konto zuzuordnen. Einer der wesentlichen Grundsätze von Universal Access besteht darin, dass kein Benutzer Zugriff auf die Daten anderer Benutzer haben darf. Wenn beispielsweise alle Anliegenaufnahmen und Screenings mithilfe nur eines einzigen Benutzerkontos (`publiccitizen`) durchgeführt würden, dann bestünde das Risiko, dass ein Benutzer Daten zu Gesicht bekommt, die von einem anderen Benutzer eingegeben wurden.

Registrierte Konten

Bei Konten dieses Typs handelt es sich um Standardkonten, die von Bürgern erstellt werden. Bürger haben die Möglichkeit, ein Konto zu erstellen, wenn sie die Anwendung erstmals aufrufen oder wenn sie Prozesse wie Screening oder Anliegenaufnahme ausführen. Diese Konten unterscheiden sich dahingehend von anonymen Konten, dass sie es Bürgern ermöglichen, zuvor gespeicherte Screenings fortzusetzen, zuvor nicht fertig gestellte Anliegenanträge erneut zu starten und zuvor übermittelte Anliegenanträge zu überprüfen oder zurückzuziehen.

Verknüpfte Konten

Bei dem letzten Kontotyp handelt es sich um verknüpfte Konten. Verknüpfte Konten sind Konten, die mit einer zugrunde liegenden ID einer Betroffenenrolle für eine Entität vom Typ `Person` in Cúram verbunden sind. Diese Benutzer haben über das Bürgerkonto Zugriff auf Detailinformationen zu ihren Leistungen und Fällen im Cúram-System. Benutzer mit einem verknüpften Konto können Lebensereignisse wie beispielsweise 'Ich habe meinen Arbeitsplatz verloren' oder 'Ich habe geheiratet' übermitteln. Darüber hinaus haben sie Zugriff auf Informationen zu Leistungszahlungen. Da diese Informationen sensibel sind, müssen Kunden sicherstellen, dass sie über einen leistungsfähigen Prozess zum Erstellen von verknüpften Benutzerkonten verfügen.

Nachstehend werden einige typische Szenarios für solche Verknüpfungen vorgestellt. Hierbei handelt es sich lediglich um Beispiele. Der tatsächliche Verknüpfungsprozess ist bei jedem Kunden unterschiedlich: Ein Kunde fordert ein Bürgerkonto an. Der Kunde wird aufgefordert, sich bei der lokalen Sozialhilfeeinrichtung mit Führerschein und anderen Ausweispapieren vorzustellen. Der Fallbearbeiter verwendet angepasste Cúram-Funktionen zum Eingeben des neuen verknüpften Kontos, nachdem die Identität des Kunden überprüft wurde.

Ein Kunde erstellt ein Benutzerkonto für Universal Access und übermittelt einen Anliegenantrag. Der Kunde wird vom Fallbearbeiter kontaktiert und gefragt, ob er weitere Services über das Universal Access-System in Anspruch nehmen möchte. Der Kunde stimmt zu und stellt sich mit entsprechenden Ausweispapieren wie beispielsweise einem Reisepass bei der örtlichen Niederlassung vor. Der Fallbearbeiter kann den Kunden mit dem Konto verknüpfen, das er zur Übermittlung des Anliegenantrags verwendet hat.

In beiden Fällen hat der Fallbearbeiter keinen Zugriff auf das Kennwort des Kunden. Stattdessen wird durch den Verknüpfungsprozess ein Stapeljob ausgelöst, der ein Schreiben generiert, das an die Privatadresse des Kunden verschickt wird. Dieses Schreiben enthält das Kennwort; ein separates Schreiben enthält eine Karte mit

einem elektronischen Code. Diese Funktionalität wird zwar gänzlich vom Kunden entwickelt, aber von den Universal Access-APIs unterstützt, die es ermöglichen, einen Universal Access-Benutzernamen mit der ID einer Betroffenenrolle zu verknüpfen.

In weiterer Folge erhält der Kunde ein Schreiben von der Sozialbehörde mit dem Anfangskennwort (im ersten Szenario). Dieses Schreiben informiert den Kunden darüber, dass eine Codekarte in Kürze folgen wird. Die Codekarte trifft am nächsten Tag per Post ein, und der Kunde kann sich daraufhin bei seinem Bürgerkonto anmelden. Die Anmeldeanzeige enthält wie gehabt einen Benutzernamen und ein Kennwort. Es gibt jedoch noch zusätzliche Authentifizierungsfaktoren: Der Kunde muss sein Geburtsdatum, seine Sozialversicherungsnummer und einen Code von der elektronischen Codekarte eingeben. Dieser Vorgang wird als Mehrfachauthentifizierung bezeichnet. Das Konzept der Mehrfachauthentifizierung wird weiter unten im vorliegenden Kapitel erläutert.

Berechtigungsrollen und -gruppen

Die vorstehend beschriebenen Kontotypen werden unterschiedlichen Berechtigungsrollen zugeordnet. Diese Rollen begrenzen die Methoden, die aufgerufen werden können. Den Berechtigungsrollen von Universal Access sollten keine zusätzlichen Berechtigungen gewährt werden. Eine Ausnahme hiervon bilden verknüpfte Konten, die die Rolle LINKEDCITIZENROLE verwenden. Werden einem Bürgerkonto zusätzliche angepasste Methoden hinzugefügt, so sind zusätzliche Berechtigungen erforderlich. Dieser Sachverhalt wird in dem Kapitel zum Thema Anpassung des Bürgerkontos erläutert.

Wenn nur eine Untergruppe der von Universal Access bereitgestellten Funktionalität verwendet wird, sollte die Berechtigung zum Aufrufen der nicht verwendeten Methoden aus der Datenbank entfernt werden. Beispiel: Wenn das Bürgerkonto nicht verwendet wird, sollten die Rolle LINKEDCITIZENROLE und andere zugehörige Artefakte entfernt werden, da sie nicht benötigt werden. Bei Projekten, in denen das Bürgerkonto nicht verwendet wird, sollten auch die Auswirkungen auf die Implementierung berücksichtigt werden. Dieser Sachverhalt wird in dem Kapitel zum Thema Anpassung des Bürgerkontos erläutert.

Berechtigungsrollen sollten nur für die Bereiche der Funktionalität konfiguriert werden, die tatsächlich verwendet werden. Es wird empfohlen, nicht verwendete SIDs aus der Datenbank zu entfernen. Beispiel: Wenn das Bürgerkonto nicht verwendet wird, sollten die Rolle LINKEDCITIZENROLE und andere zugehörige Artefakte entfernt werden, da sie nicht benötigt werden. Bei Projekten, in denen das Bürgerkonto nicht verwendet wird, sollten auch die Auswirkungen auf die Implementierung berücksichtigt werden. Weitere Informationen enthält der Abschnitt 'Bürgerkonto - Sicherheitsaspekte'.

Durch eine ordnungsgemäße Verwendung der Universal Access-Berechtigungsrollen und -Berechtigungsgruppen wird sichergestellt, dass kein Benutzer auf Funktionen zugreifen kann, für die er keine Berechtigung hat. Dadurch werden Benutzer jedoch nicht daran gehindert, diese Funktionen für den Zugriff auf Daten zu verwenden, die ihnen gehören. Dies ist Aufgabe der datenbasierten Sicherheit. Universal Access stellt ein Gerüst für datenbasierte Sicherheit bereit, das bei allen Anpassungen verwendet werden sollte. Weitere Informationen enthält der Abschnitt 'Bürgerkonto - Sicherheitsaspekte'.

Überlegungen zur Implementierung

Kundenkomponenten können in zwei Kategorien unterteilt werden: Komponenten, die zu internen Anwendungen gehören, und Komponenten, die zu öffentlich zu-

gänglichen Anwendungen (wie beispielsweise Universal Access) gehören. Komponenten mit Artefakten, die zur Verwendung in internen Anwendungen gedacht sind, sollten nicht in öffentlich zugänglichen Anwendungen wie Universal Access implementiert werden. Um dies zu erreichen, sollte bei Anpassungen zwischen internen Kundenkomponenten und öffentlich zugänglichen Kundenkomponenten unterschieden werden. Interne Komponenten sollten dem Universal Access-Implementierungspaket niemals hinzugefügt werden, da dies bedeuten würde, dass Artefakte, die für Fallbearbeiter und Administratoren bestimmt sind, in der öffentlich zugänglichen Anwendung implementiert werden.

Bei den Universal Access-Artefakten auf Kundenseite wird zwischen den Kundenkomponenten `citizenworkspace` (Citizen-Arbeitsbereich) und `citizenaccount` (Bürgerkonto) unterschieden. Dies erfolgt aus gutem Grund: Die Komponente `citizenaccount` umfasst UIM-Seiten, die sensible Daten für Bürger zugänglich machen (einschließlich der Funktionalität für Lebensereignisse), während die Komponente `citizenworkspace` die Artefakte umfasst, die für die Funktionalität für Triage, Prüfung (Screening) und Online-Antragsstellung erforderlich sind. Dementsprechend gilt: Wenn die Funktionalität des Bürgerkontos nicht verwendet wird, sollte die Kundenkomponente `citizenaccount` nicht implementiert werden, d. h. diese Komponente sollte aus dem Universal Access-Implementierungspaket entfernt werden. Weitere Informationen zur Implementierung und zum Implementierungspaket finden Sie im Handbuch zur Cúram-Implementierung für Ihren jeweiligen Anwendungsserver. Weitere Informationen zum Schutz des Bürgerkontos finden Sie im Abschnitt 'Bürgerkonto - Sicherheitsaspekte'.

Benutzernamen und Kennwörtern verwalten

Es gibt eine Reihe von Möglichkeiten, um die Validierungen, die beim Erstellen von Benutzerkonten in Universal Access aufgerufen werden, anzupassen und zu konfigurieren. Diese Methoden können verwendet werden, um bestimmte Muster für Benutzernamen und Kennwörter durchzusetzen, um beispielsweise zu verhindern, dass Benutzername und Kennwort identisch sind, oder um eine Mindestanzahl an Zeichen für Benutzernamen und/oder Kennwörter zu erzwingen.

Kontenverwaltung

In diesem Abschnitt wird beschrieben, wie die Kontoerstellung und die Kontenverwaltung angepasst werden können.

Kontenverwaltung - Konfigurationen: Mithilfe einer Reihe von Konfigurationseigenschaften wird das Verhalten der Validierungen in diesem Bereich definiert. Siehe hierzu nachstehende Tabelle:

Tabelle 10. Kontenkonfigurationen

Eigenschaft	Beschreibung
`curam.citizenworkspace.username.min.length`	Mindestanzahl der Zeichen im Benutzernamen.
`curam.citizenworkspace.password.min.length`	Mindestanzahl der Zeichen im Kennwort.
`curam.citizenworkspace.password.min.special.chars`	Mindestanzahl der Sonderzeichen und/oder Zahlen im Kennwort.

Die Werte dieser Konfigurationseigenschaften können aktualisiert werden, indem Sie sich als Systemadministrator anmelden und Folgendes auswählen: Anwendungsdaten -> Eigenschaftenadministration. Wählen Sie die Kategorie 'Citizen Self-Service - Konfiguration' aus.

Kontenverwaltung - Ereignisse: Ereignisse werden an wichtigen Punkten während der Kontoverarbeitung ausgelöst. Diese Ereignisse können verwendet werden,

um dem Kontenverwaltungsprozess angepasste Validierungen hinzuzufügen. Weitere Informationen zur Verwendung von Ereignissen finden Sie im Handbuch für Cúram-Serverentwickler (Cúram Server Developer Guide). Die folgenden Ereignisse befinden sich alle in der Klasse curam.citizenworkspace.security.impl.CitizenWorkspaceAccountEvents.

Tabelle 11. Kontoereignisse

Ereignisschnittstelle	Beschreibung
CitizenWorkspaceCreateAccountEvents	Ereignisse, die im Zusammenhang mit der Kontoerstellung ausgelöst werden. Weitere Informationen finden Sie in der zugehörigen Javadoc-Dokumentation in der Komponente 'WorkspaceServices' (Arbeitsbereichsservices).
CitizenWorkspacePasswordChangedEvent	Ereignis, das ausgelöst wird, wenn ein Benutzer sein Kennwort ändert. Weitere Informationen finden Sie in der zugehörigen Javadoc-Dokumentation in der Komponente 'WorkspaceServices' (Arbeitsbereichsservices).
CitizenWorksapceAccountAssociations	Ereignisse, die ausgelöst werden, wenn ein Benutzer mit einem zugehörigen Beteiligten vom Typ 'Person' verknüpft wird bzw. wenn eine solche Verknüpfung aufgehoben wird. Weitere Informationen finden Sie in der zugehörigen Javadoc-Dokumentation in der Komponente 'WorkspaceServices' (Arbeitsbereichsservices).

API 'CitizenWorkspaceAccountManager': Die Anwendungsprogrammierschnittstelle curam.citizenworkspace.security.impl.CitizenWorkspaceAccountManager wird verwendet, um das Erstellen und Verknüpfen von Universal Access-Konten zu verwalten. Es ist vorgesehen, dass Kunden diese API verwenden können, um angepasste Funktionen zu erstellen, die Fallbearbeiter dabei unterstützen, Konten für den Bürger zu verknüpfen und zu erstellen. Die API stellt unter anderem die folgenden Methoden zur Unterstützung der Kontenverwaltung bereit:

- Erstellen von Universal Access-Standardkonten
- Erstellen von 'verknüpften' Universal Access-Konten
- Entfernen von Verknüpfungen zwischen Beteiligten und Konten
- Abrufen von Kontoinformationen

Ausführliche Informationen hierzu enthält die Javadoc-Dokumentation zu dieser API.

Zwischenspeichern (Cashing) von Daten

Kunden müssen sich der Gefahren bewusst sein, die das Zwischenspeichern (Cashing) von Daten in Browser- und Server-Caches birgt. Es ist sorgfältig darauf zu achten, dass Bürger nicht in der Lage sind, über diese Caches auf die Daten anderer Benutzer zuzugreifen. Dieses Risiko besteht, wenn ein Bürger die Schaltfläche 'Zurück' oder den Verlauf im Browser verwendet, um Daten abzurufen, die zuvor von anderen Benutzern eingegeben wurden, oder wenn PDF-Dateien mit Anträgen lokal auf dem Computer zwischengespeichert werden, der zur Antragsstellung verwendet wurde.

HTTP-Server wie beispielsweise Apache bieten die Möglichkeit, Antwortheader zur Cachesteuerung so festzulegen, dass kein Cache gespeichert wird. Diese Methode wird auch für Universal Access-Implementierungen empfohlen, um einen Zugriff auf Daten über die Schaltfläche 'Zurück' oder den Verlauf des Browsers zu unterbinden.

Browser-Caching

Browser können so konfiguriert werden, dass Inhalte niemals im Cache zwischengespeichert werden. Wenn Universal Access in einem "Kiosk" oder anderen öffent-

lich verfügbaren Formaten angeboten werden soll, sollte der Browser so konfiguriert werden, dass Inhalte zu keiner Zeit in den Cache gestellt werden.

Darüber hinaus ist es ratsam, Universal Access so anzupassen, dass diese Anleitung auch den Bürgern gegeben wird, die über ihre eigenen Browser auf die Site zugreifen. Den Bürgern sollte empfohlen werden, den Inhalt ihres Cache zu löschen und sämtliche verwendeten Browserfenster zu schließen, nachdem sie ihre Aktivitäten in Universal Access abgeschlossen haben. Des Weiteren sollten die Bürger darüber informiert werden, dass PDF-Dateien, die sie aus Universal Access herunterladen, unter Umständen aus den temporären Internetdateien des Browsers entfernt werden müssen.

Externe Sicherheitsauthentifizierung

Mit der steigenden Präsenz staatlicher Dienste im Internet werden Anstrengungen dafür unternommen, dass die Bürger für alle Services mit einem einzigen Satz an Anmeldeinformationen authentifiziert werden können. Behörden können so den Authentifizierungsvorgang rationalisieren, während die Bürger keine langen Listen mit Benutzernamen und Kennwörtern auswendig lernen müssen. Dies wiederum erhöht die Sicherheit, da die Wahrscheinlichkeit sinkt, dass Bürger ihre Benutzernamen und Kennwörter aufschreiben, und sich die Sicherheitsbemühungen darauf konzentrieren, bewährte Verfahren in einem einzigen Unternehmenssicherheitssystem zu implementieren. Im Standardformat verwendet Universal Access ein eigenes Authentifizierungssystem, das von einer Datenbank mit registrierten Benutzern unterstützt wird. Darüber hinaus kann Universal Access für die Integration mit externen Sicherheitssystemen konfiguriert werden.

Analyse

In diesem Abschnitt wird anhand eines Beispiels die Analyse erläutert, die als Vorbereitung für die Integration mit einem externen Sicherheitssystem erforderlich ist. Bei jeder Analyse der Anforderungen für eine externe Sicherheitsintegration sollten zumindest die folgenden Fragen gestellt werden:

1. Soll die Universal Access-Implementierung anonyme Screenings und/oder anonyme Anliegen unterstützen?
2. Soll die Kontenverwaltung in Universal Access oder im externen Sicherheitssystem unterstützt werden? (Beispiel: Werden Anzeigen für die Kontoerstellung und das Zurücksetzen des Kennworts im externen Sicherheitssystem oder in Universal Access vorhanden sein?)
3. Ist die Funktion für einmalige Anmeldung (Single Sign-on) erforderlich?

Beispiel

In diesem Beispiel gibt das Implementierungsteam von Universal Access die nachstehenden Anforderungen vor. Dieses Beispiel dient als Referenz beim Beschreiben von Konfigurations- und Entwicklungstasks.

1. Benutzer können auf Universal Access zugreifen und ein anonymes Screening bzw. eine anonyme Anliegenaufnahme durchführen.
2. Benutzer, die auf ihre gespeicherten Screening- oder Anliegeninformationen zugreifen wollen, müssen zunächst ein Konto im System 'CentralID' erstellen.
3. Benutzer, die sich über die Universal Access-Anmeldeanzeige bei Universal Access anmelden, können den Benutzernamen/das Kennwort von 'CentralID' für die Authentifizierung verwenden.
4. Benutzer führen die gesamte Kontenverwaltung mithilfe eines externen Systems namens 'CentralID' aus. (Dies umfasst beispielsweise das Zurücksetzen des Kennworts, das Erstellen eines neuen Kontos und das Ändern von Kontodetails.)

5. 'CentralID' speichert alle Benutzerdatensätze auf einem sicheren LDAP-Server.
6. Da die gesamte Kontenverwaltung nun in 'CentralID' erfolgt, müssen die Anzeigen für das Erstellen von Konten und das Zurücksetzen von Kennwörtern aus Universal Access entfernt werden.
7. Benutzer sollten in der Lage sein, sich bei Universal Access anzumelden, sobald sie sich bei CentralID' registriert haben. Es sollte keine Verzögerung durch Warten auf die Weitergabe der ID an Universal Access geben.

Alle diese Anforderungen werden durch die externe Sicherheitsintegration von Universal Access unterstützt. Als diese Informationen geschrieben wurden, war die Funktion für einmalige Anmeldung (Single Sign-on) nicht Bestandteil der externen Sicherheitsintegration. Um weitere Informationen zur Unterstützung der Anforderungen für Single Sign-on zu erhalten, wenden Sie sich bitte an Cúram Global Services.

Konfigurationstasks

Auf der Grundlage des Beispiels aus dem Abschnitt zur Analyse müssen die folgenden Konfigurationstasks ausgeführt werden:

1. Konfigurieren des Anwendungsservers für die Verwendung von LDAP zur Authentifizierung.
2. Implementieren von Cúram Universal Access im Modus 'Identity Only' für registrierte Benutzer.
3. Konfigurieren von Cúram Universal Access, sodass keine Anzeigen vom Typ 'Konto erstellen' angezeigt werden.
4. Konfigurieren von Cúram Universal Access, sodass Benutzer zur Registrierung beim externen System aufgefordert werden.

Die vorstehend aufgeführten Konfigurationstasks werden in den nachstehenden Abschnitten erläutert.

Konfigurieren des Anwendungsservers für die Verwendung von LDAP zur Authentifizierung

Informationen dazu, wie Ihr Anwendungsserver für die Verwendung von LDAP zur Authentifizierung konfiguriert wird, entnehmen Sie bitte der einschlägigen Dokumentation zu diesem Server.

Implementieren von Cúram Universal Access im Modus 'Identity Only' für registrierte Benutzer

Fügen Sie der Datei 'AppServer.properties' die folgenden Eigenschaften hinzu:

```
curam.security.check.identity.only=true
curam.security.user.registry.disabled.types=EXT_AUTO,EXT_GEN
```

Um den Anwendungsserver neu zu konfigurieren, führen Sie Folgendes aus:

```
appbuild configure
```

Die Eigenschaft `curam.security.check.identity.only` stellt sicher, dass für die Anwendungssicherheit die Ausführung im Modus 'Identity Only' festgelegt ist. Weitere Informationen zum Authentifizierungsmodus 'Identity Only' finden Sie im Handbuch zur Cúram-Implementierung für WebSphere (`Cúram Deployment Guide for WebSphere`) bzw. im Handbuch zur Cúram-Implementierung für WSL (Cúram Deployment Guide for WLS). Im Modus 'Identity Only' wird für die Authentifizierung nur die interne Benutzertabelle verwendet, um zu überprüfen, ob der betreffende Benutzer vorhanden ist. Die Validierung des Kennworts wird einem nachfolgenden Modul überlassen: entweder einem JAAS-Modul (Oracle WebLogic) oder der Benutzerregistry (IBM® WebSphere).

Betrachten Sie das Beispiel eines Benutzers namens Thomas Schmidt, der beim LDAP-Server von CentralID registriert ist. Damit Thomas Schmidt Cúram Universal Access verwenden kann, muss auch ein Eintrag für Thomas Schmidt in der Tabelle für externe Benutzer (ExternalUser) vorhanden sein. Wenn sich Thomas Schmidt anmeldet, wird seine Authentifizierungsanforderung an das Cúram-JAAS-Anmeldemodul übermittelt. Dieses Modul überprüft, ob der Benutzer Thomas Schmidt in der Cúram-Tabelle 'ExternalUser' vorhanden ist, verifiziert jedoch nicht dessen Kennwort. Anschließend wechselt der Authentifizierungsprozess in die Benutzerregistry (WebSphere) oder zum LDAP-JAAS-Modul (WebLogic), wo der Benutzername und das Kennwort mit dem Inhalt des CentralID-LDAP-Servers abgeglichen werden. Damit dies ordnungsgemäß funktioniert, ist es erforderlich, den Anwendungsserver mit den Verbindungsdetails für den sicheren LDAP-Server zu konfigurieren.

Die Konfiguration von 'Identity Only' ermöglicht es der Anwendung, für die Authentifizierung der Benutzerberechtigungsnachweise auf ein externes Sicherheitssystem wie beispielsweise einen LDAP-basierten Verzeichnisservice zurückzugreifen. Dies funktioniert jedoch nicht für anonyme Benutzer von Universal Access. Wenn ein Benutzer erstmals auf die Titelseite von Universal Access zugreift, wird er automatisch als "publiccitizen" angemeldet. Wenn der Benutzer anschließend für sich ein Screening oder eine Anliegenaufnahme durchführen will, erstellt Universal Access einen neuen "generierten" anonymen Benutzer. Jeder generierte Benutzer ist eindeutig, wodurch sichergestellt wird, dass die Daten, die zu diesem Benutzer gehören, vertraulich bleiben. Weder 'publiccitizen' noch die generierten Benutzer werden in das LDAP-Verzeichnis eingefügt, sodass sie nicht mithilfe des Identity Only-Verfahrens authentifiziert werden können. Dies ist der Zweck der folgenden Konfigurationszeile:

```
curam.security.user.registry.disabled.types=EXT_AUTO,EXT_GEN
```

Mit dieser Zeile wird sichergestellt, dass Benutzer des Typs EXT_AUTO ('publiccitizen') und des Typs EXT_GEN (generierte Benutzer) anhand der Informationen in der Cúram-Tabelle für externe Benutzer authentifiziert werden. Sobald für den Server die vorstehende Konfiguration ausgeführt und der Server gestartet worden ist, müssen Sie die folgenden Konfigurationsschritt ausführen:

1. Melden Sie sich als Systemadministrator an.
2. Wählen Sie 'Anwendungsdaten -> Eigenschaftsadministration' aus.
3. Wählen Sie die Kategorie 'Bürgerkonto - Konfiguration' aus.
4. Setzen Sie die Eigenschaft 'curam.citizenaccount.public.included.user' auf den Wert EXT_AUTO.
5. Setzen Sie die Eigenschaft 'curam.citizenaccount.anonymous.included.user' auf den Wert EXT_GEN.
6. Veröffentlichen Sie die Eigenschaftsänderungen.

   Ein letzter Konfigurationseintrag ist erforderlich, um sicherzustellen, dass Universal Access im Hinblick auf die Authentifizierung ordnungsgemäß funktioniert. Die entsprechende Änderung kann wie folgt vorgenommen werden:
7. Melden Sie sich als Systemadministrator an.
8. Wählen Sie 'Anwendungsdaten -> Eigenschaftsadministration' aus.
9. Wählen Sie die Kategorie 'Infrastruktur – Sicherheitsparameter' aus.
10. Setzen Sie die Eigenschaft 'curam.custom.externalaccess.implementation' auf den Wert 'curam.citizenworkspace.security.impl.CitizenWorkspacePublicAccessSecurity'.
11. Veröffentlichen Sie die Eigenschaftsänderungen.

Abschließend müssen Sie sich abmelden und den Server erneut starten. An dieser Stelle sollte die Konfigurationstask abgeschlossen sein.

Konfigurieren von Cúram Universal Access, sodass keine Anzeigen vom Typ 'Konto erstellen' angezeigt werden

Im vorstehenden Beispiel gibt Anforderung 4 vor, dass sämtliche Kontenverwaltungsfunktionen vom externen System, CentralID, verarbeitet werden müssen. Zu diesen Funktionen gehören das Erstellen neuer Konten und das Zurücksetzen der entsprechenden Kennwörter. Universal Access stellt standardmäßig Anzeigen für diese Funktionen bereit. Diese Anzeigen müssen durch entsprechende Konfiguration wie folgt entfernt werden, um die vorstehende Anforderung 4 zu erfüllen:

1. Melden Sie sich als Systemadministrator an.
2. Wählen Sie 'Anwendungsdaten -> Eigenschaftsadministration' aus.
3. Wählen Sie die Kategorie 'Citizen Self-Service - Konfiguration' aus.
4. Setzen Sie die Eigenschaft 'curam.citizenworkspace.enable.account.creation' auf den Wert NO.
5. Veröffentlichen Sie die Eigenschaftsänderungen.

Durch die vorstehenden Schritte werden Verweise auf Seiten vom Typ 'Konto erstellen' aus Universal Access entfernt. Die Anmeldeanzeige enthält weiterhin einen Link auf eine Universal Access-Seite zum Ändern von Kennwörtern. In diesem Beispiel will das Implementierungsteam diesen Link beibehalten, aber so ändern, dass ein neues Browserfenster auf der CentralID-Seite zum Zurücksetzen des Kennworts geöffnet wird. Gehen Sie hierzu wie folgt vor:

1. Melden Sie sich als Systemadministrator an.
2. Wählen Sie 'Anwendungsdaten -> Eigenschaftsadministration' aus.
3. Wählen Sie die Kategorie 'Citizen Self-Service - Konfiguration' aus.
4. Setzen Sie die Eigenschaft 'curam.citizenworkspace.forgot.password.url' auf einen Wert wie beispielsweise 'http://www.centralid.gov/resetpassword'.
5. Veröffentlichen Sie die Eigenschaftsänderungen.

Um den Link zum Zurücksetzen des Kennworts gänzlich zu entfernen, gehen Sie wie folgt vor:

1. Melden Sie sich als Systemadministrator an.
2. Wählen Sie 'Anwendungsdaten -> Eigenschaftsadministration' aus.
3. Wählen Sie die Kategorie 'Citizen Self-Service - Konfiguration' aus.
4. Setzen Sie die Eigenschaft 'curam.citizenworkspace.display.forgot.password.link' auf den Wert NO.
5. Veröffentlichen Sie die Eigenschaftsänderungen.

Konfigurieren von Cúram Universal Access, sodass Benutzer zur Registrierung bei einem externen System aufgefordert werden

Standardmäßig fordert Universal Access die Benutzer mit folgender Nachricht auf, sich anzumelden: "Please enter your User Name and Password and click the Next button to continue." ("Geben Sie Ihren Benutzernamen und Ihr Kennwort ein und klicken Sie auf die Schaltfläche 'Weiter', um den Vorgang fortzusetzen.") Eine gute Möglichkeit, nicht registrierte Benutzer auf die CentralID-Anzeige zur Registrierung zu verweisen, besteht darin, diese Nachricht entsprechend zu ändern. So könnte die Nachricht in der Anmeldeanzeige beispielsweise in etwa wie folgt lauten:

```
"<p>If you are registered with CentralID enter your username
   and password to log in. To register go to
   <a href="http://www.centralid.gov/register"> The CentralID
   registration page.</a></p>"
```

Die Eigenschaften zum Steuern der Nachricht auf der Anmeldeseite befinden sich unter `<Cúram-Verzeichnis>/EJBServer/components/Data_Manager/Initial_Data/blob/prop/Logon.properties`.

Um die angezeigte Nachricht anzupassen, gehen Sie anhand der Schritte in Abschnitt 11.2 *Anpassbarer Inhalt von Universal Access-Seiten* in vorliegendem Handbuch vor.

Entwicklungstasks

Die bisher beschriebenen Konfigurationstasks ermöglichen es Kunden, die in dem Beispiel aufgeführten Anforderungen zu erfüllen. Eine Ausnahme hierbei bildet die folgende Anforderung:

```
"Anforderung 7 - Benutzer sollten in der Lage sein, sich bei Universal Access anzumelden,
   sobald sie sich bei 'CentralID' registriert haben; es
   sollte keine Verzögerung durch Warten auf die Weitergabe der ID an andere
   Systeme geben."
```

Für eine ordnungsgemäße Funktionsweise muss jeder Benutzer von Cúram Universal Access einen Eintrag in der Tabelle für externe Benutzer (ExternalUser) aufweisen. Der Kunde könnte einen Stapelprozess erstellen, um Benutzer aus dem LDAP-Verzeichnis in die Cúram-Tabelle 'ExternalUser' zu importieren. Dadurch würde Anforderung 7 jedoch nicht erfüllt werden, da der Benutzer in der Lage sein muss, sich bei 'CentralID' anzumelden und Universal Access anschließend unverzüglich zu verwenden. Eine andere Option wäre, einen Web-Service oder ähnlichen Mechanismus zu erstellen, der aufgerufen würde, sobald sich ein neuer Benutzer bei 'CentralID' registriert. Die Implementierung des Web-Service würde den entsprechenden Eintrag in der Tabelle 'ExternalUser' erstellen.

In vorliegendem Dokument wird nun jedoch eine einfachere Option beschrieben, die darin besteht, das standardmäßige Anmeldeverhalten zu überschreiben, um neue Konten während der Verarbeitung zu erstellen, nachdem geprüft wurde, ob der maßgebliche Eintrag im LDAP-Server vorhanden ist.

Das standardmäßige Anmeldeverhalten in Universal Access kann überschrieben werden, indem die Klasse 'curam.citizenworkspace.security.impl.SecurityStrategy' erweitert und die Methode 'authenticate()' überschrieben wird. Der nachstehende Code veranschaulicht, wie die Anwendungsprogrammierschnittstelle für die Sicherheitsstrategie (SecurityStrategy) und andere Sicherheits-APIs verwendet werden, um die vorstehend beschriebenen Anforderungen zu erfüllen:

```
public class CustomSecurityStrategy extends SecurityStrategy {
  @Inject
  private CitizenWorkspaceAccountManager cwAccountManager;
  ...
  @Override
  public String authenticate(final String username,
      final String password)
      throws AppException, InformationalException {
    final String retval = null;
    if (username.equals(PUBLIC_CITIZEN)) {
      return super.authenticate(username, password);
    }
    // Authenticate generated accounts as normal
    if (cwAccountManager.isGeneratedAccount(username)) {
      return super.authenticate(username, password);
```

```
    }
    // Check that the user exists in LDAP
    // This prevents hackers from registering a lot of bogus
    // accounts that exist in Curam but not in LDAP
    if (!isUserInLDAP(username)) {
      return SECURITYSTATUS.BADUSER;
    }
    // If there's no account for this user
    if (!cwAccountManager.hasAccount(username)) {
      createUserAccount(username);
    }
    return SECURITYSTATUS.LOGIN;
  }
  private void createUserAccount(final String username)
      throws AppException, InformationalException {
    final CreateAccountDetails newAcctDetails;
    ...
    cwAccountManager.createStandardAccount(newAcctDetails);
  }
}
```

Der vorstehende Code überprüft, ob es sich bei dem anmeldenden Benutzer um 'publiccitizen' oder ein generiertes Konto handelt. In beiden Fällen wird die Authentifizierungslogik an die Standardsicherheitsstragie delegiert. Bei einem registrierten Benutzer überprüft die Sicherheitsstrategie das LDAP-Verzeichnis, um sicherzustellen, dass der Benutzer dort vorhanden ist. Ist der Benutzer im LDAP-Verzeichnis, aber bisher nicht in Cúram vorhanden, wird ein neues Benutzerkonto erstellt. Bitte beachten Sie, dass der angepasste Code den Benutzer nicht für LDAP authentifizieren muss, da die Authentifizierung von der Benutzerregistry in Websphere bzw. vom LDAP-JAAS-Modul in WebSphere verarbeitet wird. Hierbei ist unbedingt zu beachten, dass der Kennwortparameter der Methode 'authenticate()' mithilfe einer Einweg-Hashfunktion verschlüsselt ist. Dadurch wird sichergestellt, dass das Kennwort sicher von der Clientseite der Cúram-Anwendung an die Serverseite der Anwendung übertragen werden kann.

Um die angepasste Sicherheitsstrategie (CustomSecurityStrategy) zu installieren, muss diese anstelle der Standardsicherheitsstrategie gebunden werden. Dies kann mithilfe eines Guice-Moduls zur Bindung der Implementierung erfolgen:

```
public class CustomModule extends AbstractModule {
  @Override
  protected void configure() {
    binder().bind(SecurityStrategy.class).to(
      CustomSecurityStrategy.class);
  }
}
```

Das angepasste Modul (CustomModule) muss beim Start konfiguriert werden. Dies kann erreicht werden, indem eine DMX-Datei wie folgt zur angepassten Komponente hinzugefügt wird:

```
<Cúram-Verzeichnis>/EJBServer/custom/data/initial/MODULECLASSNAME.dmx

<?xml version="1.0" encoding="UTF-8"?>
<table name="MODULECLASSNAME">
    <column name="moduleClassName" type="text" />
    <row>
      <attribute name="moduleClassName">
        <value>gov.myorg.CustomModule</value>
      </attribute>
    </row>
</table>
```

Universal Access-Triage anpassen

Einführung

In diesem Kapitel werden die Anpassungspunkte im Zusammenhang mit dem Triage-Prozess erläutert. Informationen zum Konfigurieren und Verwalten der Triage finden Sie im Handbuch zu Cúram Universal Access (`Cúram Universal Access Guide`).

Verfügbare Triage-Ereignisse

Es gibt eine Reihe von Ereignissen, die während des Triage-Prozesses ausgelöst werden. Diese Ereignisse können in zwei Kategorien eingeteilt werden: Persistenzereignisse und angepasste Ereignisse. Bei den Persistenzereignissen handelt es sich um Standardereignisse für den Zugriff auf Daten, die von der Persistenzinfrastruktur ausgelöst werden. Angepasste Ereignisse wurden hinzugefügt, um eine angepasste Verarbeitung zu ermöglichen, wenn Benutzer bestimmte Aktionen ausführen. Beide Ereignistypen werden nachstehend erläutert.

Standardmäßige Persistenzereignisse

Bei Ausführung des Triage-Regelwerks werden die Ergebnisse der Sitzung in der Entität `TriageResult` gespeichert. Dadurch wird der Aufruf der Ereignisse vor und nach der Einfügung ausgelöst. Informationen zur Verwendung der API `PersistenceEvent` für Persistenzereignisse finden Sie im `Persistence Cookbook` (Kapitel 6).

Triage-Überweisungsereignis

Das Ereignis `curam.triage.impl.TriageEvents.ReferralEvent.referralEmailSent` wird unverzüglich ausgelöst, nachdem ein Bürger sich mithilfe von Universal Access selbst für einen Service überweist. Weitere Informationen hierzu finden Sie in der API-JavaDoc-Dokumentation für 'ReferralEvent'. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/WorkspaceServices/doc`.

Universal Access-Screening anpassen

Einführung

In diesem Kapitel werden die Anpassungspunkte im Zusammenhang mit dem Prüfungsprozess (Screening) erläutert. Informationen zum Einrichten und Konfigurieren eines Screenings finden Sie in Kapitel 5 des Handbuchs zu Cúram Universal Access (`Cúram Universal Access Guide`).

Umfang, Qualität und Ergebnisse von Prüfungen überwachen

Die Klasse 'curam.citizenworkspace.impl.CWScreeningEvents' wird für den Zugriff auf die Ereignisse verwendet, die im Zusammenhang mit einer Prüfung ausgelöst wurden. Diese Funktion wird normalerweise verwenden, um den Umfang oder die Ergebnisse einer Prüfung zwecks Berichterstellung zu verfolgen. Weitere Informationen hierzu finden Sie in der API-JavaDoc-Dokumentation für 'CWScreeningEvents'. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/CitizenWorkspace/doc`.

Verarbeitung von Anliegen in Anträgen anpassen

Einführung

Den Benutzern steht eine Reihe von Anpassungspunkten zur Verfügung, auf denen aufgebaut werden kann, um den Anliegenprozess für Anträge anzupassen. In diesem Kapitel wird der Anliegenprozess bis zu dem Punkt erörtert, an dem ein Anliegenantrag übermittelt wird. Weitere Informationen zum Konfigurieren eines Anliegenantrags finden Sie im Handbuch zu Cúram Universal Access (`Cúram Universal Access Guide`).

Vorgehensweise beim Vorabausfüllen des Anliegenscripts

Das Ereignis `StartIntakeEvents.startIntake` tritt auf, bevor ein Anliegenscript ausgeführt wird. Dadurch wird ermöglicht, den Inhalt des Datenspeichers zu bearbeiten, bevor der Anliegenprozess gestartet wird. Dies erfolgt üblicherweise dann, wenn beispielsweise ein Bürger ein Screening durchlaufen hat und im Verlauf dieses Screenings einige Basisdaten hinzugefügt hat. Dieser Anpassungspunkt ermöglicht die Übertragung dieser Basisdaten an das Anliegen. Bitte beachten Sie, dass die Signatur einen Link zum Datenspeicher bereitstellt, um diesen im Voraus zu füllen. Weitere Informationen hierzu finden Sie in der API-JavaDoc-Dokumentation für `StartIntakeEvents`. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/WorkspaceServices/doc`.

Vorgehensweise beim Hinzufügen einer Validierung für Programmauswahl

Bei der Verarbeitung der vom Benutzer in der Anzeige 'Anliegenprogramm auswählen' eingegebenen Daten tritt ein Ereignis ein. Dieses Ereignis, curam.citizenworkspace.impl.ProgramSelectionEvents.intakeProgramsSelected, ermöglicht die Validierung der vom Benutzer ausgewählten Programme. Diese Funktion kann verwendet werden, um in weiterer Folge Geschäftsregeln auf einen Anliegenantrag eines Bürger anzuwenden, bei dem Produktkombinationen beispielsweise nicht kombiniert werden können. Weitere Informationen hierzu finden Sie in der API-JavaDoc-Dokumentation für `ProgramSelectionEvents`. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/CitizenWorkspace/doc`.

Bearbeitung übermittelter Anträge anpassen

Einführung

Es gibt eine Reihe von Anpassungspunkten, die während des Prozesses der Anliegenaufnahme für Anträge zur Verfügung stehen. Im vorliegenden Kapitel werden diejenigen Anpassungspunkte behandelt, die nach der Übermittlung eines Anliegenantrags vorkommen. Weitere Informationen zum Konfigurieren eines Anliegenantrags finden Sie im Handbuch zu Cúram Universal Access (`Cúram Universal Access Guide`).

Vorgehensweise bei der Anpassung der generischen PDF für verarbeitete Anträge

Universal Access stellt Funktionen bereit, um alle Anliegenanträge zu einer generischen PDF zuzuordnen, die die Werte aller vom Benutzer eingegebenen Informati-

onen aufzeichnet. Diese PDF wird vom Cúram-XML-Server bereitgestellt. Kunden haben die Möglichkeit, die Standardformatierung der generischen PDF wie folgt zu überschreiben:

Kopieren von:

- *Cúram-Verzeichnis/EJBServer/components/Workspaceservices/Data_Manager/InitialData/XSLTEMPLATEINST.dmx*

in das Verzeichnis:

- *Cúram-Verzeichnis/EJBServer/components/custom/Data_Manager/InitialData*

Bearbeiten der Datei 'project\config\datamanager_config.xml' zum Ersetzen des Eintrags für:

- *Cúram-Verzeichnis/EJBServer/components/Workspaceservices/Data_Manager/InitialData/XSLTEMPLATEINST.dmx*

durch einen Eintrag für:

- *Cúram-Verzeichnis/EJBServer/components/custom/Data_Manager/InitialData/XSLTEMPLATEINST.dmx*

Kopieren von:

- *Cúram-Verzeichnis/EJBServer/components/Workspaceservices/Data_Manager/InitialData/blob/WSXSLTEMPLATEINST001*

in das Verzeichnis:

- *Cúram-Verzeichnis/EJBServer/components/custom/Data_Manager/InitialData/blob.*

Bearbeiten dieser Datei entsprechend den Anforderungen des Projekts.

Vorgehensweise beim Verwenden von Ereignissen zur Erweiterung der Verarbeitung von Anliegenanträgen

Die Schnittstelle `IntakeApplication.IntakeApplicationEvents` enthält Ereignisse, die ausgelöst werden, nachdem der Kunde einen Anliegenantrag zur Verarbeitung übermittelt hat. Diese Ereignisse können verwendet werden, um die Art der Bearbeitung von Anliegenanträgen zu ändern, beispielsweise um die CDME-Standardzuordnung zu ergänzen oder zu ersetzen oder um eine Aktion auszuführen, nachdem ein Antrag mithilfe von Web-Services an ein fernes System gesendet wurde. Weitere Informationen hierzu finden Sie in der API-JavaDoc-Dokumentation für `IntakeApplication.IntakeApplicationEvents`. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/WorkspaceServices/doc`.

Die Schnittstelle `IntakeProgramApplication.IntakeProgramApplicationEvents` enthält Ereignisse, die in wichtigen Phasen während der Verarbeitung eines Antrags für ein bestimmtes Programm ausgelöst werden. Weitere Informationen hierzu finden Sie in der API-JavaDoc-Dokumentation für `IntakeProgramApplication.IntakeProgramApplicationEvents`. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/WorkspaceServices/doc`.

Vorgehensweise beim Senden von Anträgen zur Verarbeitung an ferne Systeme

Der Citizen-Arbeitsbereich für Bürger kann verwendet werden, um Anträge mithilfe von Web-Services zur Verarbeitung an ferne Systeme zu senden. Bevor ein Antrag an das ferne System gesendet wird, wird das Ereignis `ReceiveApplicationEvents.receiveApplication` ausgelöst. Mithilfe dieses Ereignisses kann der Inhalt des Datenspeichers, der für die Zusammenstellung der Antragsdaten verwendet wird, vor der Übermittlung bearbeitet werden. Weitere Informationen hierzu finden Sie in der API-JavaDoc-Dokumentation für `ReceiveApplicationEvents`. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/WorkspaceServices/doc`.

Vorgehensweise beim Anpassen des Workflows für die Verarbeitung von Anliegenanträgen

Kunden haben die Möglichkeit, den Workflow auf die übliche empfohlene Art und Weise anzupassen. Die entsprechende Vorgehensweise wird im Handbuch zur Cúram-Entwicklungskonformität (`Cúram Development Compliancy Guide`) und im Handbuch zum Cúram-Workflow-Management-System (`Cúram Workflow Management System Guide`) beschrieben. Hierbei ist zu beachten, dass keine Änderungen an den von diesen Workflows verwendeten Umsetzungsstrukturen vorgenommen werden sollten. Anhand dieses Workflows wird ein integrierter Fall mit dem Status 'Antrag' erstellt. Diesem integrierten Fall werden auch die Kerndaten des Falls sowie allgemeine Angabendaten zugeordnet. Dieser Prozess kann mithilfe des vorstehend beschriebenen Ereignisses 'postMapDataToCuram' abgefangen werden.

Bürgerkonto anpassen

Einführung

Das Bürgerkonto ist eine Funktion in Universal Access, die es einem verknüpften Universal Access-Benutzer ermöglicht, sich in einem sicheren Bereich anzumelden, in dem er Programme prüfen und beantragen kann. Ebenso kann der Bürger für ihn relevante Informationen anzeigen lassen, darunter individuell zugeschnittene Nachrichten, systemweite Ankündigungen, Aktualisierungen seiner Zahlungen, Kontaktinformationen von Behördenmitarbeitern oder Outreach-Kampagnen, die für ihn von Interesse sein könnten. Darüber hinaus bietet das Bürgerkonto ein Gerüst für Kunden, um eigene Bürgerkontoseiten zu erstellen oder die vorhandenen Seiten zu überschreiben.

Das Handbuch zu Cúram Universal Access (`Cúram Universal Access Guide`) enthält eine vollständige Beschreibung der gesamten angebotenen Standardfunktionalität sowie weitere Informationen zu verknüpften Universal Access-Benutzern.

Technische Übersicht

Im Gegensatz zum Rest der Universal Access-Anwendung wird das Gerüst des Bürgerkontos im Benutzerschnittstellenmanager (UIM) definiert. Dies bedeutet, dass Kunden vorhandene Seiten überschreiben, ihre eigenen Seiten hinzufügen und die Navigation des Gerüsts anpassen können, ebenso wie es möglich ist, die Fallbearbeiteranwendung anzupassen.

Das Bürgerkonto baut auf der Benutzerschnittstelleninfrastruktur auf. Es verwendet lediglich eine Untergruppe der von der Infrastruktur gebotenen Benutzer-

schnittstellen- und Navigationskomponenten, um eine einfache, verwendbare Anwendung bereitzustellen, die von den Bürgern verstanden und ohne besondere Schulung genutzt werden kann.

Verknüpfte Universal Access-Benutzer können über ihr Konto die Universal Access-Aktionen Triage, Prüfung (Screening) und Online-Antragstellung durchführen. Dementsprechend umfasst das Bürgerkonto UIM-Seiten, die eine Sicht auf die Funktionen für Triage, Prüfung und Online-Antragstellung bieten. Diese Seiten können weder konfiguriert noch angepasst werden: Die von ihnen bereitgestellte Funktionalität kann über die Administration konfiguriert werden. Entsprechende Informationen hierzu enthält die Dokumentation zu diesen Bereichen. Die UIM-Seiten in Zusammenhang mit Triage, Prüfung und Online-Antragstellung sollten weder geändert noch überschrieben werden.

Sicherheitsaspekte

Die Bereitstellung sensibler Daten für Bürger über das World Wide Web ist naturgemäß gefährlich, und die Datensicherheit muss bei der Entwicklung von Anpassungen für Bürgerkonten oberste Priorität haben. Bitte lesen Sie das Kapitel zum Thema `Schutz von Universal Access`, um weitere Informationen hierzu zu erhalten. Es wird dringend empfohlen, alle öffentlich zugänglichen Anwendungen vor der Implementierung einer strengen Sicherheitsanalyse und strengen Sicherheitstests zu unterziehen. Darüber hinaus wird empfohlen, Unterstützung anfordern, um außergewöhnliche Anpassungen zu erörtern, die möglicherweise spezielle Sicherheitsrisiken bergen.

Die Berechtigung zum Aufrufen der Serverfassadenmethoden, die Daten an Bürgerkontoseiten liefern, wird vom standardmäßigen Berechtigungsmodell verwaltet. Weitere Informationen finden Sie im Handbuch für Cúram-Serverentwickler (`Curám Server Developer's Guide`). Zusätzlich zu den standardmäßigen Berechtigungsprüfungen muss jede Fassadenmethode, die von einer Bürgerkontoseite aufgerufen wird, die folgenden Sicherheitsprüfungen durchführen, um sicherzustellen, dass der Benutzer, der einer Transaktion zugeordnet ist (der momentan angemeldete Benutzer) über die Berechtigung verfügt, auf die angeforderten Daten zuzugreifen:

- Es muss sichergestellt werden, dass der momentan angemeldete Benutzer den korrekten Typ aufweist. Es muss sich um einen externen Benutzer mit dem Anwendungscode CITWSAPP und einem Universal Access-Konto vom Typ 'Verknüpft' handeln.
- Es muss sichergestellt werden, dass der momentan angemeldete Benutzer über die Berechtigung verfügt, auf die von ihm gelesenen Datensätze zuzugreifen, d. h. es müssen alle übergebenen Seitenparameter validiert werden, um sicherzustellen, dass die angeforderten Datensätze auf die eine oder andere Weise tatsächlich mit dem momentan angemeldeten Benutzer in Zusammenhang stehen.

Korrekten Typ des momentan angemeldeten Benutzers sicherstellen

Die Anwendungsprogrammierschnittstelle `curam.citizenaccount.security.impl.CitizenAccountSecurity` stellt die Methode 'performDefaultSecurityChecks' bereit, mit der sichergestellt werden kann, dass der Benutzer den korrekten Typ aufweist. Diese Methode überprüft den Benutzertyp. Ist der Benutzertyp unzulässig, wird eine Nachricht in die Protokolle geschrieben, und die Transaktion schlägt fehl. *Diese Methode sollte in der ersten Zeile jeder angepassten Fassadenmethode aufgerufen werden, bevor irgendwelche Verarbeitungsschritte oder weitere Validierungen stattgefunden haben:*

```
public CitizenPaymentInstDetailsList listCitizenPayments()
    throws AppException, InformationalException {

    // perform security checks
    citizenAccountSecurity.performDefaultSecurityChecks();

    // validate any page parameters (none in this case)

    // invoke business logic
    return citizenPayments.listPayments();
  }
```

Zugriffsberechtigung des momentan angemeldeten Benutzers auf speziell angeforderte Datensätze sicherstellen

Ein heimtückischer Benutzer, der bei einem gültigen verknüpften Universal Access-Konto angemeldet ist, könnte Anforderungen zum Abrufen von Daten zu anderen Benutzern an das System senden. Um dies zu verhindern, müssen sämtliche Seitenparameter validiert werden, um sicherzustellen, das sie auf die eine oder andere Weise tracefähig sind und zum momentan angemeldeten Benutzer zurückverfolgt werden können. Die entsprechende Vorgehensweise hängt vom jeweiligen Datensatztyp ab. Eine Zahlung beispielsweise kann über den Fall, für den sie ausgegeben wurde, zum Beteiligten zurückverfolgt werden.

Die Anwendungsprogrammierschnittstelle `curam.citizenaccount.security.impl.CitizenAccountSecurity` bietet Methoden, mit denen solche Prüfungen für die Datensatztypen durchgeführt werden können, die den Bürgern über die OOTB-Seiten geliefert werden. Spezielle Informationen hierzu finden Sie in der Javadoc-Dokumentation zu dieser API. Bei angepassten Seiten, die unterschiedliche Arten von Daten liefern, müssen zusätzliche Prüfungen implementiert werden, um die Seitenparameter zu validieren. Diese Prüfungen sollten zu einer angepassten Sicherheits-API hinzugefügt und von den betreffenden Fassadenmethoden aufgerufen werden. Die Methoden sollten überprüfen, ob der angeforderte Datensatz zum momentan angemeldeten Benutzer zurückverfolgt werden kann, und im gegenteiligen Fall den Benutzernamen, den Methodennamen sowie weitere Daten protokollieren und die Transaktion unverzüglich abbrechen (anstatt das Problem dem Validierungshilfsprogramm hinzuzufügen und die Transaktion fortfahren zu lassen):

```
if (paymentInstrument.getConcernRole().getID()
   != citizenWorkspaceAccountManager
     .getLoggedInUserConcernRoleID().getID()) {

 /**
 * the payment instrument passed in is not related
 * to the logged in user log the user name of the
 * current user, the method invoked and any other
 * pertinent data
 */

 // throw a generic message
 throw PUBLICUSERSECURITYExceptionCreator
   .ERR_CITIZEN_WORKSPACE_UNAUTHORISED_METHOD_INVOKATION();
 }
```

Wenngleich so viele Informationen wie möglich zu einem Übergriff protokolliert werden sollten, muss unbedingt sichergestellt werden, dass die ausgelösten Ausnahmebedingungen keine Informationen preisgeben, die heimtückischen Benutzern von Nutzen sein könnten. Es sollte eine generische Ausnahmebedingung ausgelöst werden, die keine Informationen dazu enthält, warum die Transaktion fehlgeschlagen ist. Die Anwendungsprogrammierschnittstelle `curam.citizenaccount.security.impl.CitizenAccountSecurity` gibt eine generische

Nachricht aus, die darauf hinweist, dass der betreffende Benutzer nicht berechtigt ist, auf die angeforderte Seite zuzugreifen.

Vorgehensweise beim Hinzufügen einer neuen Seite zum Bürgerkonto

In diesem Abschnitt werden die Tasks erläutert, die ausgeführt werden müssen, um eine angepasste Seite zu einem Bürgerkonto hinzuzufügen.

Angepasste, externe Kundenkomponente erstellen

Artefakte, die zu Anwendungen gehören, die an die Öffentlichkeit gerichtet sind (so wie Universal Access), sollten in separaten Komponenten gespeichert werden, um zu verhindern, dass interne Seiten, die für Administratoren und Fallbearbeiter bestimmt sind, in Anwendungen implementiert werden, die an die Öffentlichkeit gerichtet sind. Folglich besteht der erste Schritt beim Hinzufügen einer angepassten Seite zu einem Bürgerkonto darin, eine neue, angepasste kundenseitige Komponente einzurichten, in die diese Seite gestellt werden soll. Anweisungen zur entsprechenden Vorgehensweise finden Sie im Abschnitt zum Thema 'Externe Kundenanwendungen' im Handbuch für Serverentwickler (Server Developer Guide). Diese Komponente sollte diejenigen Artefakte enthalten, die für Universal Access implementiert werden sollen, und keine Artefakte, die für Administratoren oder Fallbearbeiter gedacht sind. Diese Komponente muss dem Implementierungspaket für die Universal Access-EAR-Datei hinzugefügt werden. Wichtig hierbei ist, dass die Datei 'deployment_packaging.xml' im Ordner `<Curam-Verzeichnis>/EJBServer/project/config/` die Mindesteinträge für die Komponenten enthält, die erstellt werden müssen, d. h. 'CitizenAccount', 'CitizenWorkspace', 'IntelligentEvidenceGathering' und 'CEFWidgets'.

UIM-Seite in neuer Komponente erstellen

Entwickeln Sie die angepasste UIM-Seite in der neuen Kundenkomponente. Die Infrastruktur der Benutzerschnittstelle bietet ein umfassendes Sortiment an komplexen Funktionen. Bitte bedenken Sie hierbei, dass die Zielgruppe dieser Seite Bürger sind, die sich wahrscheinlich nicht mit komplexen Benutzerschnittstellen auskennen, sodass es ratsam ist, Bürgerkontoseiten relativ einfach zu gestalten, im Gegensatz zu den komplexen Benutzerschnittstellen, die für erfahrene Benutzer wie beispielsweise Fallbearbeiter und Administratoren entwickelt werden.

Navigationseintrag für neue Seite hinzufügen

Für diesen Vorgang wird die Standardmethode verwendet. Informationen zur Vorgehensweise beim Erweitern der Navigation entnehmen Sie bitte dem Handbuch für Cúram-Schnittstellenentwickler (`Cúram User Interface Developers Guide`).

```
<nc:navigation id="CitizenAccount">
    <nc:nodes>
      <nc:navigation-page
        id="home" page-id="CitizenAccount_certification"
          title="leaf.title.certification" />
    </nc:nodes>
  </nc:navigation>
```

Abbildung 1. Beispiel eines angepassten Navigationseintrags für angepasste Bürgerkontoseite

Fassade erstellen

Entwickeln Sie eine Fassade, die von der Seite aufgerufen werden kann. Diese Fassade ruft Daten entweder auf Grundlage des momentan angemeldeten Benutzers oder auf Grundlage von übergebenen Seitenparametern ab. Im Allgemeinen lesen Bürgerkontoseiten Daten im Zusammenhang mit den verknüpften Konten des angemeldeten Benutzers. Insbesondere dann, wenn der angemeldete Benutzer mit ei-

nem Cúram-Beteiligten verknüpft ist (d. h. mit der ID der Rolle eines Betroffenen, concernRoleID), werden Daten im Zusammenhang mit der Rolle dieses Betroffenen, den relevanten Fällen und den entsprechenden Angaben angezeigt. Wenn der Benutzer mit fernen Fallverarbeitungssystemen verknüpft ist, können Daten aus diesen fernen Systemen im Bürgerkonto angezeigt werden. Die Anwendungsprogrammierschnittstelle `curam.citizenworkspace.security.impl.CitizenWorkspaceAccountManager` bietet eine Methode zur Vereinfachung, die verwendet werden kann, um die verknüpften Identitäten eines momentan angemeldeten Benutzers, einschließlich der verknüpften Rolle des Betroffenen (sofern vorhanden), abzurufen. Kunden wird empfohlen, diese Anwendungsprogrammierschnittstelle (API) zu verwenden, um verknüpfte Identitäten abzurufen, da diese API integrierte Sicherheitsprüfungen umfasst, um sicherzustellen, dass der fragliche Benutzer tatsächlich ein verknüpfter Universal Access-Benutzer ist.

Es müssen entsprechende Autorisierungseinträge in DMX hinzugefügt werden, um den verknüpften Universal Access-Benutzern die Berechtigung zu erteilen, die neue Fassadenmethode aufzurufen. Fügen Sie einen Eintrag für die neue Methode zur Gruppe LINKEDCITIZENWORKSPACEGROUP hinzu. Beispiel:

```
<row>
     <attribute name="groupName">
       <value>LINKEDCITIZENWORKSPACEGROUP</value>
     </attribute>
     <attribute name="sidName">
       <value>MyCustomFacadeName.myCustomFacadeMethodName</value>
     </attribute>
   </row>
```

Vorgehensweise beim Anpassen von Universal Access-Formatvorlagen in Bürgerkonten

Universal Access-Formatvorlagen für Bürgerkonten können mithilfe der Standardmethode für UIM-Seiten angepasst werden. Informationen hierzu finden Sie im Handbuch für Cúram-Benutzerschnittstellenentwickler (`Cúram User Interface Developers Guide`). Die Formatvorlagen für Bürgerkonten befinden sich im Verzeichnis `webclient/components/CitizenAccount/css`.

Ländereinstellungen anpassen

Hierbei ist zu beachten, dass die Methode zum Hinzufügen neuer Ländereinstellungen zu Bürgerkonten der Methode für UIM-Standardseiten entspricht, im Gegensatz zur dynamischen Methode, mit der öffentliche Universal Access-Seiten international verwendbar gemacht werden können. Um verschiedene Ländereinstellungen zu Bürgerkonten hinzuzufügen, muss das Clientprojekt erneut erstellt werden, um die JSPs in den betreffenden neuen Ländereinstellungen zu generieren. Weitere Informationen zum Verwalten von UIM-Seiten, die in mehreren Ländereinstellungen angeboten werden sollen, finden Sie im Handbuch für Cúram-Web-Client-Entwickler (Cúram Web Client Developer Guide). Die Codetabelle CT_APPLICATION_CODE wird verwendet, um Anwendungscodes externer Benutzer der jeweiligen UIM-Seite zuzuordnen, an die die Benutzer beim Anmelden weitergeleitet werden sollen. Die Clientinfrastruktur verwendet diese Konfigurationen, um zu ermitteln, wohin ein Benutzer nach erfolgreicher Authentifizierung weitergeleitet werden soll. Universal Access wird mit einem Eintrag für die standardmäßige Ländereinstellung "en" geliefert:

```
<code default="false" java_identifier="CITIZEN_WORKSPACE"
 status="ENABLED" value="CITWSAPP">
  <locale language="en" sort_order="0">
    <description>startseite_des_bürgerkontos</description>
  </locale>
</code>
```

Beim Hinzufügen weiterer Ländereinstellungen zu Universal Access muss für jede dieser Ländereinstellungen ein zusätzlicher Eintrag in diese Codetabelle eingefügt werden. Alle Einträge sollten dieselbe Beschreibung enthalten, die den Wert für die Startseite des Bürgerkontos angibt. Dies gilt auch für andere Codetabellen, die von Universal Access verwendet werden, wie beispielsweise 'CT_SecretQuestionType'. Die Namen der Geheimfragen müssen jeder Ländereinstellung hinzugefügt werden, für die Universal Access angeboten werden soll. Weitere Informationen zur Lokalisierung finden Sie im Kapitel zu den vollständig anpassbaren Self-Service-Artefakten.

Startseite des Bürgerkontos

Die Startseite eines Bürgerkontos bietet eine Reihe von Funktionen für die Bereitstellung relevanter Informationen für die Bürger. Weitere Informationen hierzu finden Sie im Handbuch zu Cúram Universal Access (`Cúram Universal Access Guide`). Die verschiedenen Bereiche werden mithilfe unterschiedlicher Methoden konfiguriert, die im Handbuch zur Cúram Universal Access-Konfiguration (`Cúram Universal Access Configuration Guide` beschrieben werden.

Wenn in diesem Handbuch vom Anpassen der Startseite des Bürgerkontos die Rede ist, bezieht sich dies auf die Bestandteile dieser Seite, die Anzeigen für Outreach (Wussten Sie?) und 'Meine Nachrichten'.

Anzeigetext anpassen

Die Willkommensnachricht, die Clustertitel und der Text der Nachricht bei der letzten Anmeldung werden in einer Eigenschaftendatei im Ressourcenspeicher gespeichert. Der Name des angemeldeten Benutzers wird an die Eigenschaft `citizenaccount.welcome.caption` angehängt, um die Willkommensnachricht auf der Startseite anzuzeigen. Um die Willkommensnachricht zu ändern, muss diese Eigenschaft in der Eigenschaftendatei entsprechend angepasst werden. Die Datei befindet sich im Verzeichnis `EJBServer\components\CitizenWorkspace\data\initial\blob\prop\CitizenMessageMyPayments.properties`. Der Text kann angepasst werden, indem eine neue Version mit einem Namensattribut "CitizenAccountHome" in den Ressourcenspeicher hochgeladen wird.

Outreach-Kampagnen

Outreach-Kampagnen werden entwickelt, um zielgruppenspezifische Kampagnennachrichten für den Bürger anzuzeigen. Diese Nachrichten können Bilder sowie Links zu Universal Access-Seiten und externen Websites enthalten. Ob eine spezielle Kampagne für einen bestimmten Bürger angezeigt werden soll oder nicht, wird von einem CER-Regelwerk festgelegt, das der entsprechenden Kampagne in der Administration zugeordnet wird. Outreach wird mithilfe von Advisor und CER entwickelt, und die Ausgabe von Kampagnen wird als Advice-Posten aufgezeichnet. Weitere Informationen enthält das Handbuch zur Cúram Universal Access-Konfiguration (`Cúram Universal Access Configuration Guide`.

Vorgehensweise beim Konfigurieren einer neuen Bürgerkampagne: Informationen zum Konfigurieren von neuen Outreach-Kampagnen in der Administration finden Sie im Handbuch zur Universal Access-Konfiguration (`Universal Access Configuration Guide`.

Regelwerk für Outreach-Kampagnen: Outreach baut auf der Advisor-Infrastruktur auf, und das Regelwerk 'CoreCitizenCampaignRuleset', das alle Outreach-Kampagnen erweitern sollten, übernimmt wiederum Regeln aus dem Regelwerk 'CoreAdvisorRuleset'. Das Regelwerk 'CoreCitizenCampaignRuleset' befindet sich im Verzeichnis '/EJBServer/components/citizenworkspace/CREOLE_Rule_Sets'.

Das Regelwerk 'CoreCitizenCampaignRuleset' definiert zwei Regelklassen, die zur Steuerung von Outreach-Kampagnen verwendet werden:

Regelklasse 'CitizenCampaignAdmin'

Hierbei handelt es sich um eine CER-Regeldarstellung eines CitizenCampaign-Verwaltungsdatensatzes. Der Name, das Ablaufdatum und die Ablaufuhrzeit sowie der Bildverweis für eine Kampagne werden weitergegeben. Ein Regelobjekt dieser Klasse ist für jede aktive Outreach-Kampagne im System vorhanden. Die Verwaltung erfolgt intern durch die Outreach-Infrastruktur.

Regelklasse 'AbstractCampaignAdviceItem'

Diese Klasse erweitert 'AbstractAdviceItem' (siehe hierzu die Advisor-Dokumentation). Diese Klasse muss von Regelklassen konkreter Outreach-Kampagnen erweitert werden. In Regelklassen konkreter Outreach-Kampagnen müssen die folgenden Attribute angegeben werden, die von dieser Regelklasse übernommen werden:

- citizenCampaignName

  Namen sind kampagnenübergreifend eindeutig, da sie als eindeutige Kennungen verwendet werden. Das im Regelwerk angegebene Attribut 'citizenCampaignName' und der Name der Outreach-Kampagne bei deren Erstellung in der Administration *müssen identisch sein*. Daher muss der Name der in der Administration erstellten neuen Outreach-Kampagne mit dem Attribut 'citizenCampaignName' übereinstimmen, das im zugehörigen Regelwerk angegeben ist.
- campaignShowAdvice

  In diesem Attribut befindet sich die Geschäftslogik der Kampagne. Hier muss der Wert 'true' zurückgegeben werden, wenn der betreffende Beteiligte die Kriterien zum Anzeigen der Kampagne erfüllt.

Die Klasse 'AbstractCampaignAdviceItem' setzt das Attribut 'showAdvice' der zugehörigen übergeordneten Klasse in Abhängigkeit davon, ob ein CitizenCampaign-Admin-Regelobjekt für die betreffende Kampagne vorhanden ist (d. h. die Kampagne also in Administration aktiv ist), und in Abhängigkeit des Wertes für das Attribut 'campaignShowAdvice'.

Standardmäßig werden Ablaufdatum und -uhrzeit einer Kampagne dem Verwaltungsdatensatz der Outreach-Kampagne entnommen. Dies ermöglicht es Administratoren, den Ablauftermin von Kampagnen zu konfigurieren. Es ist jedoch bei Bedarf auch möglich, Ablaufdatum und -uhrzeit auf Basis der Geschäftslogik oder anderer Regeln festzulegen, indem das Attribut 'expiryDateTime' der Klasse 'AbstractCampaignAdviceItem' in der entsprechenden untergeordneten Implementierung dieser Klasse überschrieben wird.

In den Regelklassen konkreter Kampagnen muss auch eine Klasse deklariert werden, die den Advisor-Kontext 'AbstractAdviceContext' erweitert. Weitere Informationen enthalten die Advisor-Dokumentation und das folgende Beispielregelwerk für Kampagnen.

```
<RuleSet name="SampleCampaignRuleSet">

 <!-- This class is infrastructure used by Advisor,
   please refer to the Advisor documentation for more
   information. -->

 <Class extends="AbstractAdviceContext"
   extendsRuleSet="CoreAdvisorRuleSet"
     name="SampleCampaignContext">

   <!-- populated by advisor propagator -->
   <Attribute name="concernRoleID">
     <type>
       <ruleclass name="NumberParameter"
         ruleset="CoreAdvisorRuleSet"/>
     </type>
     <derivation>
       <specified/>
     </derivation>
   </Attribute>

   <!-- populated by advisor propagator -->
   <Attribute name="adviceContextID">
     <type>
       <javaclass name="Number"/>
     </type>
     <derivation>
       <specified/>
     </derivation>
   </Attribute>

   <Attribute name="advice">
     <type>
       <javaclass name="List">
         <ruleclass name="AbstractCampaignAdviceItem"
           ruleset="CoreCitizenCampaignRuleset"/>
       </javaclass>
     </type>
     <derivation>
       <fixedlist>
         <listof>
           <ruleclass name="AbstractCampaignAdviceItem"
             ruleset="CoreCitizenCampaignRuleset"/>
         </listof>
         <members>
         <!-- This list of members must include the custom rule
                 class that extends AbstractCampaignAdviceItem -->
           <create ruleclass="SampleCampaign">
             <this/>
           </create>
         </members>
       </fixedlist>
     </derivation>
   </Attribute>
 </Class>

 <!-- Concrete Campaign / Advisor class that extends
         AbstractCampaignAdviceItem -->
 <Class extends="AbstractCampaignAdviceItem"
   extendsRuleSet="CoreCitizenCampaignRuleset"
     name="SampleCampaign">

   <!-- initialise the Advisor context. Please see Advisor
           documentation for more information -->
   <Initialization>
     <Attribute name="sampleCampaignContext">
       <type>
```

```
        <ruleclass name="SampleCampaignContext"/>
      </type>
    </Attribute>
  </Initialization>

  <!-- This is a reference to the campaign text stored in the
          resource store. Please see the Advisor documentation
          for more information. -->
  <Attribute name="adviceText">
    <type>
      <javaclass name="String"/>
    </type>
    <derivation>
      <String value="propertyName"/>
    </derivation>
  </Attribute>

  <!-- This is a reference to the advice context ID.
       Please see the Advisor documentation for more
       information. -->
  <Attribute name="adviceContext">
    <type>
      <javaclass name="Number"/>
    </type>
    <derivation>
      <reference attribute="adviceContextID">
        <reference attribute="sampleCampaignContext"/>
      </reference>
    </derivation>
  </Attribute>

  <!-- This is used by the parent abstract class to read the
       campaign rule object. This name must be identical to
       the name given to the Outreach campaign in
       Administration -->
  <Attribute name="citizenCampaignName">
    <type>
      <javaclass name="String"/>
    </type>
    <derivation>
      <String value="SampleCampaign"/>
    </derivation>
  </Attribute>

  <!-- Whether or not to display the campaign for the given
          participant (provided the campaign is Active) -->
  <Attribute name="campaignShowAdvice">
    <type>
      <javaclass name="Boolean"/>
    </type>
    <derivation>
      <!-- business logic for campaign goes here. -->
      <true/>
    </derivation>
  </Attribute>
 </Class>
</RuleSet>
```

Bilder und Links: Der Advisor und somit auch Outreach unterstützen das Einbinden von Bildern und Links in ein Advice-Element bzw. eine Kampagne. Das Bild selbst wird beim Erstellen einer neuen Outreach-Kampagne hochgeladen. Standardmäßig wird ein Bild, das beim Erstellen der Kampagne in der Administration angegeben wird, als Teil der Kampagne ohne einen Link angezeigt. Es ist jedoch

auch möglich, einen Link im Regelwerk anzugeben und in diesem Link wiederum ein Bild anzugeben, um so auf das für die Kampagne konfigurierte Bild zu verweisen.

Definieren Sie im Regelwerk der angepassten konkreten Kampagne einen Link:

```
<Class extends="AbstractLink"
   extendsRuleSet="CoreAdvisorRuleSet"
     name="ChildCareOptionLinkWithImage">

   <Attribute name="name">
     <type>
       <javaclass name="String"/>
     </type>
     <derivation>
       <String value="childCareOptionLinkImage"/>
     </derivation>
   </Attribute>

   <Attribute name="target">
     <type>
       <javaclass name="String"/>
     </type>
     <derivation>
       <String value="http://www.yourtargeturl.com"/>
     </derivation>
   </Attribute>

   <Attribute name="modal">
     <type>
       <javaclass name="Boolean"/>
     </type>
     <derivation>
       <false/>
     </derivation>
   </Attribute>

   <Attribute name="external">
     <type>
       <javaclass name="Boolean"/>
     </type>
     <derivation>
       <true/>
     </derivation>
   </Attribute>

   <Attribute name="linkImage">
     <type>
       <ruleclass name="Image" ruleset="CoreAdvisorRuleSet"/>
     </type>
     <derivation>
     <!-- note that this is specified. The parent rule class will
             specify the image reference from the campaign -->
       <specified/>
     </derivation>
   </Attribute>

 </Class>
```

Wenn Sie diesen Link in der angepassten Implementierung von AbstractCampaignAdviceItem deklarieren, geben Sie den Verweis auf das in der Administration konfigurierte Bild an. Weitere Informationen zum Definieren von Links und Bildern finden Sie in der Dokumentation zum Advisor.

```
<Attribute name="childCareOptionLinkWithImage">
   <type>
     <ruleclass name="ChildCareOptionLinkWithImage"/>
   </type>
   <derivation>
     <create ruleclass="ChildCareOptionLinkWithImage">
       <specify attribute="linkImage">
         <reference attribute="campaignImage"/>
       </specify>
     </create>
   </derivation>
 </Attribute>
```

Damit die Links im Zusammenhang mit imageOnly-Kampagnen in den Advisor-Datenbanktabellen gespeichert (und somit in Outreach-Kampagnen angezeigt) werden, ist ein Eintrag in der Eigenschaftendatei der betreffenden Kampagne erforderlich. Beispiel:

```
AdviceItem.imageOnlyText={link::imageCampaignLinkWithImage}
```

Dieser Eintrag gibt keinen Namen für den Link an, sondern verweist auf den Namen des Regelobjekts, das im Regelwerk der Kampagne definiert ist, um diesen Link darzustellen.

Leistungsaspekte: Kampagnen verweisen auf CER-Regelobjekte, um zu bestimmen, ob Kampagnen angezeigt werden sollen oder nicht. Wenn sich die zugrunde liegenden Daten, von denen diese Regelobjekte abhängen, ändern, wird daher eine CER-Neubewertung ausgelöst. Daraufhin wird vom Advisor neu berechnet, ob die Kampagne angezeigt werden soll oder nicht. Dies kann sich auf die Leistung auswirken und muss entsprechend berücksichtigt werden. Hierbei kann es sich um zwei Arten von Datenänderungen handeln:

Änderungen an den Daten der Beteiligten

Diese Art Änderungen wirken sich auf einen bestimmten Beteiligten aus. Betrachten Sie beispielsweise eine Kampagne, die auf die Adresse eines Bürgers verweist. Sobald ein Benutzer seine Adresse ändert, wird diese Änderung an das Regelobjekt weitergegeben, das die Adresse dieses Beteiligten darstellt. Da das Regelobjekt der Kampagne hiervon abhängt, wird auch eine Neubewertung ausgelöst. Dies bedeutet, dass bei jeder Adressänderung des Beteiligten die Kampagnenregeln ausgeführt werden, um zu ermitteln, ob die betreffende Kampagne weiterhin angezeigt werden soll oder nicht. Aus diesem Grund ist es wichtig zu berücksichtigen, wie häufig eine Änderung an einem Datenelement vorkommen kann, für wie viele Bürger dies gilt und ob ein Verweis auf dieses Datenelement in einer Kampagne zu Leistungsproblemen im System führen kann.

Änderungen an Outreach-Kampagnen in der Administration

Diese Art Änderungen wirken sich auf alle Regelausführungen aus, die mit der betreffenden Kampagne in Zusammenhang stehen. Dies bedeutet, dass eine Neubewertung für jeden Bürger ausgelöst wird, der gemäß einer vorherigen Bewertung für diese Kampagne anspruchsberechtigt ist. Wenn beispielsweise das einer Kampagne zugeordnete Bild geändert wird, führt das System die Regeln für jeden Bürger erneut aus, der für diese Kampagne berücksichtigt worden ist. Hierfür ist unter Umständen ein hoher Verarbeitungsaufwand erforderlich, der zu einem Leistungsproblem im System führen könnte. Daher wird empfohlen, Änderungen in der Kampagnenadministration zu einem Zeitpunkt durchzuführen, an dem das System keine Lastspitzen aufweist oder zu Wartungszwecken in den Offlinemodus versetzt wurde.

Meine Nachrichten

Das Handbuch zu Cúram Universal Access (`Cúram Universal Access Guide`) enthält eine Übersicht über die von der Nachrichtenanzeige gebotene Funktionalität sowie über die jeweils standardmäßig bereitgestellten Nachrichten.

Wenn sich ein verknüpfter Benutzer anmeldet, werden Nachrichten aus der Systemumgebung sowie aus fernen Systemen für die Anzeige zusammengestellt. Diese Arbeit wird von der Anwendungsprogrammierschnittstelle `curam.citizenmessages.impl.CitizenMessageController` übernommen. Sie liest die in der Datenbanktabelle 'ParticipantMessage' gespeicherten Nachrichten nach Beteiligten und löst auch das Ereignis 'CitizenMessagesEvent.userRequestsMessages' aus, um Listener aufzufordern, Nachrichten zu einer Liste hinzuzufügen, die als Teil des Ereignisparameters übergeben wird. Die aus den einzelnen Quellen zusammengestellten Nachrichten werden sortiert, in XML umgewandelt und zum Anzeigen an den Kunden zurückgegeben.

Bürgernachrichten konfigurieren: Es gibt globale Konfigurationen, die für Bürgernachrichten angegeben werden können. Hierzu gehören beispielsweise die Aktivierung bestimmter Typen und die Konfiguration ihrer Anzeigereihenfolge. Die verschiedenen Nachrichtentypen umfassen darüber hinaus auch eigene Konfigurationspunkte. Spezielle Informationen hinsichtlich der Vorgehensweise beim Anpassen der verschiedenen Nachrichtentypen finden Sie in nachfolgenden Abschnitten im vorliegenden Dokument. Weitere Informationen zur Vorgehensweise beim Ändern der globalen Konfigurationen und beim Bereitstellen von Bürgernachrichten mithilfe von Web-Services finden Sie im Handbuch zur Cúram Universal Access-Konfiguration (`Cúram Universal Access Configuration Guide`).

Der Textinhalt eines Nachrichtentyps kann ebenfalls konfiguriert werden. Für jeden Nachrichtentyp gibt es eine zugehörige Eigenschaftendatei, in der die lokalisierbaren Texteinträge für die verschiedenen Nachrichten enthalten sind, die für den betreffenden Typ angezeigt werden. Diese Eigenschaften umfassen auch Platzhalter, die zur Laufzeit durch reale Werte im Zusammenhang mit dem Bürger ersetzt werden.

Die Formulierung dieser Texte kann angepasst werden, indem eine andere Version der Eigenschaftendatei in den Ressourcenspeicher eingefügt wird. Die nachstehende Tabelle enthält Angaben dazu, welche Eigenschaftendatei für die einzelnen Nachrichtentypen jeweils geändert werden muss:

Tabelle 12. Eigenschaftendateien für Nachrichten

Nachrichtentyp	Name der Eigenschaftendatei
Zahlungen	CitizenMessageMyPayments.properties
Antragsbestätigung	CitizenMessageApplicationAcknowledgement.properties
Verifizierungen	CitizenMessageVerificationMessages.properties
Besprechungen	CitizenMessageMeetingMessages.properties
Überweisung	CitizenMessagesReferral.properties
Servicebereitstellung	CitizenMessagesServiceDelivery.properties

Es besteht auch die Möglichkeit, Platzhalter (die zur Laufzeit mit aktuellen Daten gefüllt werden) aus den Eigenschaften zu entfernen. Derzeit ist es jedoch nicht möglich, zusätzliche Platzhalter zu bestehenden Nachrichten hinzuzufügen. In einem solchen Fall muss ein angepasster Nachrichtentyp implementiert werden.

Neuen Typ Bürgernachricht hinzufügen: Nachrichten werden vom Controller mithilfe mithilfe von zwei Methoden zusammengestellt: Der Controller liest Nachrichten, die in der Datenbank gespeichert wurden, über die Anwendungsprogrammierschnittstelle `curam.citizenmessages.persistence.impl.ParticipantMessage` und stellt Nachrichten auch durch Auslösen des Ereignisses `curam.participantmessages.events.impl.CitizenMessagesEvent` zusammen.

Es muss entschieden werden, ob die Nachrichten mit einer Push-Operation an die Datenbank übertragen werden sollen oder ob sie während der Verarbeitung dynamisch von einem Listener generiert werden sollen, der für das Ereignis empfangsbereit ist, das ausgelöst wird, wenn sich der Bürger anmeldet. Die speziellen Anforderungen des Nachrichtentyps müssen hierbei ebenso berücksichtigt werden wie die Vorteile und Nachteile der jeweiligen Option.

Gespeicherte Nachrichten

In diesem Szenario wird immer dann eine Nachricht in der Datenbank gespeichert, wenn im System etwas stattfindet, das möglicherweise für den Bürger von Interesse ist. Wird beispielsweise eine Einladung zu einer Besprechung erstellt, wird ein Ereignis ausgelöst. Die OOTB-Funktionalität für Besprechungsnachrichten ist für dieses Ereignis empfangsbereit, und wenn die zur Besprechung eingeladene Person ein Beteiligter mit einem verknüpften Universal Access-Konto ist, wird eine Nachricht an die Tabelle 'ParticipantMessage' geschrieben, die den Bürger darüber informiert, das eine Einladung zu der Besprechung vorliegt.

Ein Vorteil dieser Methode besteht darin, dass der Verarbeitungsaufwand gering ist, wenn sich der Bürger anmeldet, um diese Nachricht anzuzeigen: Die Nachricht wird einfach aus der Datenbank gelesen und angezeigt. Es findet keine Berechnung statt, um zu ermitteln, ob die Nachricht erforderlich ist oder nicht. Allerdings muss die Implementierung auch alle Änderungen an den zugrunde liegenden Daten verarbeiten, die die Nachricht möglicherweise ungültig machen oder modifizieren könnten, und entsprechende Maßnahmen ausführen. Beispielsweise ist die Funktionalität für Besprechungsnachrichten auch für Änderungen an Besprechungen empfangsbereit, um sicherzustellen, dass die Besprechungszeit, der Besprechungsort etc. aktuell sind, und um gegebenenfalls eine neue Nachricht an den Bürger auszugeben, um ihn darüber zu informieren, dass sich Ort oder Zeit geändert haben.

Dynamische Nachrichten

Diese Nachrichten werden von Ereignislistenern generiert, wenn sich der Bürger anmeldet. Diese Listener sind für das Ereignis `curam.participantmessages.events.impl.CitizenMessagesEvent.userRequestsMessages` empfangsbereit.

Ein Vorteil besteht darin, dass kein Code für die Verwaltung von Änderungen im zeitlichen Verlauf erforderlich ist, da die Nachricht zur Laufzeit generiert wird. Die Nachricht wird auf der Grundlage der Daten im System jedesmal generiert, wenn sich der Bürger anmeldet. Sollten sich also Änderungen an den zugrunde liegenden Daten ergeben, erhält der Bürger bei der nächsten Anmeldung dennoch die korrekte Nachricht.

Ein Nachteil dieser Methode liegt darin, dass während der Laufzeit unter Umständen ein deutlicher Verarbeitungsaufwand anfällt, um die Nachricht zu generieren. Daher ist darauf zu achten, dass diese Methode nicht die Ladezeit der Startseite des Bürgerkontos beeinträchtigt.

Leistungsaspekte müssen gegen den erforderlichen Aufwand für die Verwaltung von Änderungen an den Daten im Zusammenhang mit der Nachricht im zeitlichen Verlauf sowie gegen die Anforderungen des jeweiligen Nachrichtentyps abgewogen werden. Beispiel: Die OOTB-Verifizierungsnachricht ist dynamisch. Wenn sich ein Bürger anmeldet, wird geprüft, ob für den betreffenden Bürger noch ausstehende Prüfungen vorliegen. Hierbei handelt es sich um einen relativ einfachen Datenbanklesevorgang, wohingegen es komplizierter wäre, für zahlreiche Ereignisse in der Verifizierungsengine empfangsbereit zu sein und sicherzustellen, dass im Hinblick auf die ausstehenden Prüfungen des Beteiligten eine aktuelle Nachricht in der Datenbank gespeichert wurde. Auf der anderen Seite müssen die Besprechungsnachrichten den Bürger über Änderungen an den relevanten Besprechungen informieren, sodass entsprechende Funktionen geschrieben werden mussten, um Änderungen am Besprechungsdatensatz und an der zugehörigen Nachricht im zeitlichen Verlauf verwalten zu können.

Neuen Nachrichtentyp implementieren: Um einen neuen Nachrichtentyp zu implementieren, müssen die folgenden Schritte ausgeführt werden, und zwar unabhängig davon, ob die Nachricht gespeichert oder dynamisch generiert wird:

Allgemeine Tasks

- Fügen Sie der Codetabelle `CT_ParticipantMessageType` einen neuen Eintrag hinzu, um den Nachrichtentyp darzustellen. Dieser Eintrag wird in der Administration verwendet, um den neuen Nachrichtentyp zu konfigurieren.
- Fügen Sie einen DMX-Eintrag für die Datenbanktabelle 'ParticipantMessageConfig' hinzu. In diesem Eintrag, der für die Administration verwendet wird, werden der Typ und die Sortierreihenfolge des neuen Nachrichtentyps gespeichert. Beispiel:

```
<row>
   <attribute name="PARTICIPANTMESSAGECONFIGID">
     <value>2110</value>
   </attribute>
   <attribute name="PARTICIPANTMESSAGETYPE">
     <value>PMT2001</value>
   </attribute>
   <attribute name="ENABLEDIND">
     <value>1</value>
   </attribute>
   <attribute name="SORTORDER">
     <value>5</value>
   </attribute>
   <attribute name="VERSIONNO">
     <value>1</value>
   </attribute>
 </row>
```

- Fügen Sie dem Anwendungsressourcenspeicher eine Eigenschaftendatei hinzu, die die Texteigenschaften und Bildverweise für die Nachricht enthält.
- Fügen Sie dem Ressourcenspeicher ein Bild für diese Nachricht hinzu.

Dynamische Nachricht implementieren

Um eine dynamische Nachricht implementieren zu können, muss ein Ereignislistener implementiert werden, der für das Ereignis `CitzenMessagesEvent.userRequestsMessages` empfangsbereit ist. Dieses Ereignisargument enthält einen Verweis auf den Beteiligten und eine Liste, zu der der Listener Java-Objekte vom Typ `curam.participantmessages.impl.ParticipantMessage` hinzufügt. Weitere Informationen hierzu enthält die Javacoc-API-Dokumentation für `CitzenMessagesEvent`. Diese Dokumentation befindet sich im Verzeichnis `<Cúram-Verzeichnis>/EJBServer/components/core/doc`.

Entwickler sollten auch die Javadoc-API-Dokumentation für curam.participantmessages.impl.ParticipantMessage und curam.participantmessages.impl.ParticipantMessages zu Rate ziehen, um eine vollständige Erläuterung zu erhalten.

Der Nachrichtentext befindet sich in einer Eigenschaftendatei im Ressourcenspeicher. Ein dynamischer Listener ruft die relevanten Eigenschaften aus dem Ressourcenspeicher ab und erstellt entsprechend das Objekt 'ParticipantMessage'. Der Nachrichtentext für eine bestimmte Nachricht kann Platzhalter umfassen. Werte für Platzhalter werden als Parameter zu Objekten vom Typ 'ParticipantMessage' hinzugefügt. Der Controller für Bürgernachrichten (CitizenMessagesController) löst diese Platzhalter auf, indem er sie durch die tatsächlichen Werte im Zusammenhang mit dem fraglichen Beteiligten ersetzt, die dem Nachrichtenobjekt als Parameter hinzugefügt wurden.

Betrachten Sie beispielsweise folgenden Eintrag in der Datei 'CitizenMessage-MyPayment.properties':

```
Message.First.Payment=
  Your next payment is due on {Payment.Due.Date}
```

Die tatsächliche Fälligkeit der betreffenden Zahlung wird dem Objekt 'ParticipantMessage' als Parameter hinzugefügt (siehe nachstehenden Beispielcode). Der CitizenMessagesController löst die Platzhalter daraufhin auf, indem der Text mit den tatsächlichen Werten gefüllt wird, und wandelt die Nachricht anschließend in XML um. Diese Nachricht wird dann auf der Startseite des Bürgerkontos ausgegeben. (Es gibt auch eine öffentliche Methode 'CitizenMessageController', die alle Nachrichten für einen Bürger als Liste zurückgibt. Weitere Informationen hierzu enthält die Javadoc-Dokumentation.)

Vorgehensweise über die Anwendungsprogrammierschnittstelle 'curam.participantmessages.impl.ParticipantMessage':

```
/**
 * Adds a parameter to the map. The paramReference
 * should be present in the message title or body so
 * it can be replaced by the paramValue before the message
 * is displayed.
 *
 * @param paramReference
 * a string place holder that is present in either the
 * message title or body. Used to indicate where the value
 * parameter should be positioned in a message.

 * @param paramValue
 * the value to be substituted in place of the place holder
 */
 public void addParameter(final String paramReference,
   final String paramValue) {

   parameters.put(paramReference, paramValue);
 }
```

Der Aufruf an die Methode würde wie folgt aussehen:

```
participantMessage.addParameter("Payment.Due.Date", "1/1/2011");
```

Nachrichten können auch Links enthalten. Ebenso wie Platzhalter werden Links zur Laufzeit aufgelöst. In Links können Platzhalterwerte als Text verwendet werden, der für den betreffenden Link angezeigt werden soll. Ein Link wird in einer Eigenschaftendatei wie folgt definiert:

```
Click {link:here:paymentDetails} to view the payment details.
```

In diesem Beispiel ist "here" der anzuzeigende Text, und "paymentDetails" verweist auf den Namen des Links, der an diesem Punkt in den Text eingefügt werden soll. Weitere Informationen finden Sie im Handbuch für Advisor-Entwickler (`Advisor Developer's Guide`). Damit dieser Link von einem dynamischen Listener mit einem Ziel gefüllt wird, würde der Listener ein Objekt vom Typ `curam.participantmessages.impl.ParticipantMessageLink` erstellen und ein Ziel und einen Namen für den Link angeben. Der Code würde wie folgt aussehen:

```
ParicipantMessageLink participantMessageLink =
     new ParticipantMessageLink(false,
       "CitizenAccount_listPayments", "paymentDetails");

   participantMessage.addLink(participantMessageLink);
```

Bevor die Nachricht erstellt wird, muss der dynamische Listener durch Prüfung sicherstellen, dass der fragliche Nachrichtentyp momentan aktiviert ist. Der Datensatz `curam.participantmessages.configuration.impl.ParticipantMessageConfiguration` für diesen Nachrichtentyp muss gelesen werden und die Methode `isEnabled` verwendet werden, um zu ermitteln, ob dieser Nachrichtentyp aktiviert ist. Ist dies nicht der Fall, findet keine weitere Verarbeitung statt.

* Es wird empfohlen, den Code, der für das Ereignis empfangsbereit ist, und den Code, der eine bestimmte Nachricht erstellt, voneinander zu trennen, um der Philosophie zu folgen, dass 'jeder das tut, was er am besten kann'.

Gespeicherte Nachricht implementieren

Damit dem Bürger eine gespeicherte Nachricht angezeigt werden kann, muss diese Nachricht über die Anwendungsprogrammierschnittstelle `curam.citizenmessages.persistence.impl.ParticipantMessage` in die Datenbank geschrieben werden. Nachrichtenargumente werden verarbeitet, indem ein Datensatz vom Typ `curam.advisor.impl.Parameter` gespeichert und dem Datensatz 'ParticipantMessage' zugeordnet wird. Die Verknüpfung erfolgt über die API `curam.advisor.impl.Link`. Parameternamen sollten den im Nachrichtentext enthaltenen Platzhaltern zugeordnet werden. Linknamen sollten sich auf die Namen von Links beziehen, die im Nachrichtentext angegeben sind. Die Javadoc-Dokumentation zu `curam.citizenmessages.persistence.impl.ParticipantMessage`, `curam.advisor.impl.Parameter` und `curam.advisor.impl.Link` enthält weitere Informationen hierzu.

Für jede Beteiligtennachricht muss ein Ablaufdatum angegeben werden. Nach Ablauf dieses Datums wird die Nachricht nicht mehr angezeigt.

Nachrichten können aus der Datenbank entfernt werden. Wenn eine Nachricht durch eine geänderte Version ersetzt werden muss oder aus einem anderen Grund entfernt werden muss, kann dies über die Anwendungsprogrammierschnittstelle `curam.citizenmessages.persistence.impl.ParticipantMessage` vorgenommen werden.

Jeder Nachricht sind eine ID und ein Typ zugeordnet. Diese werden verwendet, um den Datensatz, der mit der Nachricht in Zusammenhang steht, zurückverfolgen zu können. Besprechungsnachrichten beispielsweise werden mit der ID 'Aktivität' und dem Typ 'Besprechung' gespeichert. Nachrichten können über `ParticipantMessageDAO` nach Beteiligtem, zugehöriger ID und Typ gelesen werden.

Bevor die Nachricht gespeichert wird, muss der dynamische Listener durch Prüfung sicherstellen, dass der fragliche Nachrichtentyp momentan aktiviert ist. Der Datensatz
curam.participantmessages.configuration.impl.ParticipantMessageConfiguration
für diesen Nachrichtentyp muss gelesen werden und die Methode isEnabled verwendet werden, um zu ermitteln, ob dieser Nachrichtentyp aktiviert ist. Ist dies nicht der Fall, findet keine weitere Verarbeitung statt.

Spezielle Nachrichtentypen anpassen: Die von OOTB bereitgestellten Nachrichtentypen können auf verschiedene Art und Weise angepasst werden. Die entsprechenden Methoden werden in diesem Abschnitt beschrieben. Eine Beschreibung der unterschiedlichen Nachrichtentypen finden Sie im Handbuch zu Cúram Universal Access (Cúram Universal Access Guide).

Nachricht für Überweisung

Mit diesem Nachrichtentyp werden Nachrichten im Zusammenhang mit Überweisungen erstellt. Hierbei handelt es sich um dynamische Nachrichten. Wenn sich ein Bürger anmeldet, wird für jede für diesen Bürger im System vorhandene Überweisung eine Nachricht erstellt, sofern die Überweisung das aktuelle Datum oder ein Datum in der Zukunft aufweist und sofern für diese Überweisung ein entsprechendes Serviceangebot angegeben ist. Die Eigenschaftendatei EJBServer\components\CitizenWorkspace\data\initial\blob\prop\CitizenMessageReferral.properties enthält die Eigenschaften für den Nachrichtentext der Überweisung, die Nachrichtenparameter, die Links und die Bilder. Diese Eigenschaftendatei wird im Ressourcenspeicher abgelegt. Diese Ressource wird unter dem Ressourcennamen CitizenMessageReferral registriert. Um den Text der Nachricht zu ändern, Platzhalter zu entfernen oder Links zu ändern, muss eine neue Version dieser Datei in den Ressourcenspeicher hochgeladen werden.

Nachricht zu Servicebereitstellung

Mit diesem Nachrichtentyp werden Nachrichten im Zusammenhang mit der Bereitstellung von Services erstellt. Hierbei handelt es sich um dynamische Nachrichten. Wenn sich ein Bürger anmeldet, wird für jede für diesen Bürger im System vorhandene Servicebereitstellung eine Nachricht erstellt, sofern die Servicebereitstellung den Status 'In Bearbeitung' oder 'Nicht gestartet' aufweist. Die Eigenschaftendatei EJBServer\components\CitizenWorkspace\data\initial\blob\prop\CitizenMessageServiceDelivery.properties enthält die Eigenschaften für den Nachrichtentext für die Servicebereitstellung, die Nachrichtenparameter, die Links und die Bilder. Diese Eigenschaftendatei wird im Ressourcenspeicher abgelegt. Diese Ressource wird unter dem Ressourcennamen CitizenMessageServiceDelivery registriert. Um den Text der Nachricht zu ändern, Platzhalter zu entfernen oder Links zu ändern, muss eine neue Version dieser Datei in den Ressourcenspeicher hochgeladen werden.

Zahlungsmeldungen: Mit diesem Nachrichtentyp werden Nachrichten auf Basis der Zahlungsausgänge, abgebrochenen Zahlungen etc. für einen Bürger erstellt. Diese Nachrichten werden in der Datenbank gespeichert. Vorhandene Nachrichten werden durch neue ersetzt, d. h. wenn eine Zahlung beispielsweise ausgegeben und anschließend abgebrochen wird, wird die Nachricht über den Zahlungsausgang durch eine Nachricht über die abgebrochene Zahlung ersetzt. Die Eigenschaftendatei EJBServer\components\CitizenWorkspace\data\initial\blob\prop\CitizenMessageMyPayments.properties enthält die Eigenschaften für den Text der Finanznachricht, die Nachrichtenparameter, die Links und die Bilder. Diese Eigenschaftendatei wird im Ressourcenspeicher abgelegt. Diese Ressource wird unter

dem Ressourcennamen CitizenMessageMyPayments registriert. Um den Text von Finanznachrichten zu ändern, Platzhalter zu entfernen oder Links zu ändern, muss eine neue Version dieser Datei in den Ressourcenspeicher hochgeladen werden. Die nachstehende Tabelle beschreibt die Nachrichten, die erstellt werden, wenn unterschiedliche Ereignisse im Zusammenhang mit Zahlungen im System vorkommen, sowie die Eigenschaften in der Datei CitizenMessageMyPayments.properties, die mit den erstellten Nachrichten jeweils in Zusammenhang stehen.

Tabelle 13. Zahlungsnachrichten und zugehörige Eigenschaften

Zahlungsereignis	Nachrichteneigenschaft
Erste für einen Fall ausgegebene Zahlung	Message.First.Payment
Neueste ausgegebene Zahlung	Message.Payment.Latest
Letzte ausgegebene Zahlung	Message.Last.Payment
Abgebrochene Zahlung	Message.Cancelled.Payment
Erneut ausgegebene Zahlung	Message.Reissue.Payment
Gestoppte Zahlung (Fall unterbrochen)	Message.Stopped.Payment
Unterbrechung der Zahlung / des Falls aufgehoben	Message.Unsuspended.Payment

Anpassung des Ablaufdatums für Zahlungsnachrichten

Mithilfe einer Systemeigenschaft kann konfiguriert werden, wie viele Tage die Zahlungsnachricht dem Bürger angezeigt werden soll. Standardmäßig ist der Eigenschaftswert auf 10 Tage gesetzt. Dieser Wert kann jedoch über die Eigenschafsadministration überschrieben werden.

Tabelle 14. Eigenschaft für Ablauf der Zahlungsnachricht

Name	Beschreibung
curam.citizenaccount.payment.message.expiry.days	Die Zeitdauer in Tagen, über die die Zahlungsnachricht dem Beteiligten angezeigt wird.

Besprechungsnachrichten: Mit diesem Nachrichtentyp werden Nachrichten auf der Grundlage von Besprechungen erstellt, zu denen der Bürger eingeladen ist. Voraussetzung hierfür ist, dass die Besprechungen über die Anwendungsprogrammierschnittstelle curam.meetings.sl.impl.Meeting erstellt werden. Diese Anwendungsprogrammierschnittstelle löst Ereignisse aus, die von der Funktion für Besprechungsnachrichten verarbeitet werden. Es gibt auch andere Möglichkeiten, Aktivitätsdatensätze ohne diese Anwendungsprogrammierschnittstelle zu erstellen. Allerdings werden für solche Besprechungen keine zugehörigen Besprechungsnachrichten erstellt, da die betreffenden Ereignisse nicht ausgelöst werden. Diese Nachrichten werden in der Datenbank gespeichert. Vorhandene Nachrichten werden durch neue ersetzt, d. h. wenn eine Besprechung terminiert ist und sich der Besprechungsort anschließend ändert, wird die ursprüngliche Einladung durch eine Nachricht ersetzt, die den Bürger über den Ortswechsel informiert. Die Eigenschaftendatei EJBServer\components\CitizenWorkspace\data\initial\blob\prop\CitizenMessageMeetingMessages.properties enthält die Eigenschaften für den Besprechungsnachrichtentext, die Nachrichtenparameter, die Links und die Bilder. Diese Eigenschaftendatei wird im Ressourcenspeicher abgelegt. Diese Ressource wird unter dem Ressourcennamen CitizenMessageMeetingMessages registriert. Um den Text von Besprechungsnachrichten zu ändern, Platzhalter zu entfernen oder Links zu ändern, muss eine neue Version dieser Datei in den Ressourcenspeicher hochgeladen werden. Die nachstehende Tabelle beschreibt die Nachrichten, die erstellt werden, wenn unterschiedliche Ereignisse im Zusammenhang mit Besprechungen im System vorkommen, sowie die Eigenschaften in der Datei CitizenMessageMeetingMessages.properties, die mit den erstellten Nachrichten jeweils in Zusammenhang stehen. In Abhängigkeit davon, ob es sich um eine ganztätige Be-

sprechung handelt, ob ein Besprechungsort angegeben wurde und ob die Person, die die Besprechung vorbereitet, Kontaktinformationen im System registriert hat, werden unterschiedliche Version des Nachrichtentexts angezeigt. Dementsprechend handelt es sich bei den Eigenschaftswerten in dieser Tabelle um Näherungswerte, die sich auf eine Reihe von Eigenschaften in der Eigenschaftendatei beziehen. Eine vollständige Liste der Nachrichteneigenschaften finden Sie in der Eigenschaftendatei selbst.

Tabelle 15. Besprechungsnachrichten

Besprechungsereignis	Nachrichteneigenschaften
Besprechungseinladung	Non.Allday.Meeting.Invitation.*, Allday.Meeting.Invitation.*
Besprechungsaktualisierung	Non.Allday.Meeting.Update.*, Allday.Meeting.Update.*
Besprechungsstornierung	Allday.Meeting.Update.*, Allday.Meeting.Cancellation.*

Anpassung des Anzeigedatums für Besprechungsnachrichten

Mithilfe einer Systemeigenschaft kann konfiguriert werden, wie viele Tage vor dem Startdatum der Besprechung die Nachricht dem Bürger angezeigt werden soll. Standardmäßig ist der Eigenschaftswert auf 10 Tage gesetzt. Dieser Wert kann jedoch über die Eigenschafsadministration überschrieben werden.

Die Besprechungsnachricht läuft am Ende der Besprechung, d. h. zum geplanten Endtermin (Datum/Uhrzeit) der Besprechung, ab. Nach Ablauf wird die Nachricht dem Bürger nicht mehr angezeigt.

Tabelle 16. Eigenschaft für Anzeigedatum von Besprechungsnachrichten

Name	Beschreibung
curam.citizenaccount.meeting.message.effective.days	Gibt an, wie viele Tage vor dem Startdatum der Besprechung die Nachricht dem Bürger angezeigt werden soll.

Anpassung der Aktivitätstypen, für die Besprechungsnachrichten erstellt werden sollen

Besprechungen werden in der Aktivitätsinstanz gespeichert. Es gibt verschiedene Aktivitätstypen, die in der Codetabelle `CT_ActivityType` gespeichert sind. Die Liste der Aktivitätstypen, für die Nachrichten erstellt werden sollen, kann mithilfe der folgenden Eigenschaft angepasst werden. Der Standardcode lautet 'AT2'. Dieser Code steht für Besprechung.

Tabelle 17. Aktivitätstypen, für die Besprechungsnachrichten erstellt werden sollen

Name	Beschreibung
curam.citizenaccount.meeting. activity.types.to.generate.messages	Eine Konfigurationseinstellung zur Vorgabe der Aktivitätstypen, für die Nachrichten erstellt werden.

Antragsbestätigungsnachricht: Bei diesem Nachrichtentyp wird eine Nachricht erstellt, wenn ein Antrag von einem Bürger übermittelt wird. Diese Nachricht wird in der Datenbank gespeichert. Die Eigenschaftendatei `EJBServer\components\CitizenWorkspace\data\initial\blob\prop\CitizenMessageApplicationAcknowledgment.properties` enthält die Eigenschaften für den Nachrichtentext, die Nachrichtenparameter, die Links und die Bilder. Diese Eigenschaftendatei wird im Ressourcenspeicher abgelegt. Diese Ressource wird unter dem Ressourcennamen `CitizenMessageApplicationAcknowledgment` regist-

riert. Um den Text der Nachricht zu ändern, Platzhalter zu entfernen oder Links zu ändern, muss eine neue Version dieser Datei in den Ressourcenspeicher hochgeladen werden.

Anpassung des Ablaufdatums für Antragsbestätigungsnachricht

Die Zeitdauer in Tagen, über die die Antragsbestätigungsnachricht dem Bürger angezeigt wird, kann mithilfe einer Systemeigenschaft konfiguriert werden. Standardmäßig ist der Eigenschaftswert auf 10 Tage gesetzt. Dieser Wert kann jedoch über die Eigenschafsadministration überschrieben werden.

Tabelle 18. Eigenschaft für Ablauf der Antragsbestätigungsnachricht

Name	Beschreibung
curam.citizenaccount. intake.application.acknowledgement.message.expiry.days	Die Zeitdauer in Tagen, über die die Antragsbestätigungsnachricht dem Beteiligten angezeigt wird.

Vorhandene Seiten anpassen

Die mit OOTB gelieferten Seiten 'Meine Zahlungen', 'Meine Anträge' und 'Meine Aktivitäten' können angepasst werden. Wie UIM-Standardseiten können auch die UIM-Seiten höher in der Komponentenreihenfolge ersetzt werden, und es können entsprechende Änderungen vorgenommen werden.

Auf der Serverseite können die Anwendungsprogrammierschnittstellen (APIs), die diese Seiten steuern, angepasst werden; sie befinden sich im Paket `curam.citizenaccount.impl`. Sie können zusammen mit den Strukturen, die sie zurückgeben, angepasst werden, um zusätzliche Informationen zu liefern. Diese Art der Anpassung ist der Anpassung der Fassade `curam.citizenaccount.facade.impl.CitizenAccount` vorzuziehen, da diese Fassade intern ist und nicht aufgerufen werden sollte.

Die erforderlichen Sicherheitsprüfungen befinden sich in der Fassade `CitizenAccount` und nicht in den APIs, die sich in `curam.citizenaccount.impl` befinden. Angepasste Fassaden müssen die erforderlichen Prüfungen implementieren.

Die Strukturen, die zur Lieferung von Daten an diese Seiten verwendet werden, können ebenfalls angepasst werden. Die Modelldateien, die diese Strukturen enthalten, befinden sich in `EJBServer\components\CitizenWorkspace\model\Packages\CitizenAccount`. Im Gegensatz zu anderen Modellartefakten in Universal Access können diese Dateien anhand der Standardmethode angepasst werden.

Anpassung der Seite 'Meine Zahlungen'

Daten für diese Seite werden mithilfe der Anwendungsprogrammierschnittstelle `curam.citizenaccount.impl.CitizenPayments` abgerufen. Mithilfe der Methode `listPayments` werden die Zahlungen auf der Seite aufgelistet. Die integrierte Seite für Anweisungsdetails ruft die Methode `readPaymentInstructionByInstrument` auf, um Zahlungsanweisungsdetails abzurufen.

Anpassung der Seite 'Meine Anträge'

Daten für diese Seite werden mithilfe der Anwendungsprogrammierschnittstelle `curam.citizenaccount.impl.CitizenPayments` abgerufen. Mithilfe der Methode `listPayments` werden die Zahlungen auf der Seite aufgelistet. Die integrierte Seite für Anweisungsdetails ruft die Methode `readPaymentInstructionByInstrument` auf, um Zahlungsanweisungsdetails abzurufen. Bei Verwendung dieser Anwendungsprogrammierschnittstelle in angepassten Fassaden sind hierbei die erforderlichen Sicherheitsprüfungen zu berücksichtigen.

Anpassung der Seite 'Kontaktinformationen'

Auf dieser Seite werden die Kontaktinformationen des Fallbearbeiters für jeden Fall im Zusammenhang mit dem Bürger sowie die Kontaktinformationen des betreffenden Bürgers angezeigt. Diese Seite kann auf verschiedene Art und Weise angepasst werden. Informationen dazu enthält die nachstehende Tabelle.

Tabelle 19. Eigenschaften zum Anpassen von Kontaktinformationen

Name	Beschreibung	Standardwert
curam.citizenaccount. contactinformation.show.caseworker.details	Gibt an, ob die Kontaktinformationen des Fallbearbeiters auf dieser Seite angezeigt werden sollen oder nicht.	true (wahr)
curam.citizenaccount. contactinformation.show.casemember.cases	Gibt an, ob die Details des Fallbearbeiters für die Fälle angezeigt werden sollen, bei denen der Bürger ein Fallmitglied und nicht der Primärbeteiligte ist.	true (wahr)
curam.citizenaccount. contactinformation.show.businessphone	Gibt an, ob die Geschäftstelefonnummer des Fallbearbeiters angezeigt werden soll oder nicht.	true (wahr)
curam.citizenaccount. contactinformation.show.mobilephone	Gibt an, ob die Mobiltelefonnummer des Fallbearbeiters angezeigt werden soll oder nicht.	true (wahr)
curam.citizenaccount. contactinformation.show.faxnumber	Gibt an, ob die Faxnummer des Fallbearbeiters angezeigt werden soll oder nicht.	true (wahr)
curam.citizenaccount. contactinformation.show.pagernumber	Gibt an, ob die Pagernummer des Fallbearbeiters angezeigt werden soll oder nicht.	true (wahr)
curam.citizenaccount. contactinformation.show.emailaddress	Gibt an, ob die E-Mail-Adresse des Fallbearbeiters angezeigt werden soll oder nicht.	true (wahr)

Lebensereignisse anpassen

Zweck

Das vorliegende Kapitel dient dem folgenden Zweck:

- Erläuterung, was ein Lebensereignis ist und warum Lebensereignisse nützlich sind.
- Erläuterung, wie einfache Lebensereignisse entwickelt werden.

Zielgruppe

Das vorliegende Kapitel richtet sich an Geschäftsanalysten, Softwarearchitekten und Entwickler. Viele Typen von Lebensereignissen können gänzlich von Analysten (Bearbeitern) erstellt werden, während für bestimmte Typen die Hilfe von Entwicklern erforderlich ist. Die Informationen in diesem Kapitel sollen Analysten ein besseres Verständnis davon vermitteln, wie die Analyse für ein neues Lebensereignis durchgeführt wird und wie festgestellt werden kann, ob die Hilfe von Entwicklern benötigt wird.

Übersicht

Das vorliegende Kapitel soll dem Leser einen Überblick über das Thema Lebensereignisse bieten und eine vollständige Anleitung zur Entwicklung solcher Lebensereignisse an die Hand geben.

Abschnitt 1 bietet eine kurze Einführung in das Thema Lebensereignisse.

Abschnitt 2 beschreibt die allgemeine Architektur von Lebensereignissen und erörtert, wie die Analyse- und Entwicklungstasks beim Erstellen eines Lebensereignisses ausgeführt werden.

Abschnitt 3 beschreibt die Java-Anwendungsprogrammierschnittstelle für einfache Lebensereignisse.

Lebensereignisse - Einführung

Lebensereignisse sind dazu gedacht, eine ganzheitliche Sicht davon zu erfassen, was im Leben einer Person vorgeht. Dabei liefern Lebensereignisse nicht nur unaufbereitete Informationen zu den Lebensumständen, dem Einkommen und ähnlichen Aspekten einer Person, sondern stellen diese auch in einem Kontext dar.

Betrachten Sie folgendes Szenario: Peter Schmidt hat seinen Arbeitsplatz verloren, nachdem die Firma, für die er zuvor gearbeitet hatte, geschlossen wurde. Peter meldet sich bei seinem Bürgerkonto an und ruft den Abschnitt für Lebensereignisse auf. Er wählt das Lebensereignis 'Arbeitsplatz verloren' aus. Das System startet ein IEG2-Script, um Informationen zum Ereignis 'Arbeitsplatzverlust' zu erfassen. Das Script stellt Peter eine Reihe relevanter Fragen zu den Umständen seines Arbeitsplatzverlusts. Diese Fragen sind nicht unbedingt für bestimmte soziale Unterstützungsprogramme relevant, an denen Peter unter Umständen teilnimmt. Ein Lebensereignisscript ist üblicherweise kurz und prägnant. Es ist möglich, dass eventuell zu Peter Schmidt bekannte Informationen bereits vorab im Script vorhanden sind. So kann es beispielsweise sein, dass der Name und die Adresse von Peters früherem Arbeitgeber im Script angezeigt werden (dies wird als Vorabausfüllen des IEG-Scripts bezeichnet). Peter bestätigt, dass es sich tatsächlich um den Arbeitgeber handelt, der ihn entlassen hat.

Nachdem das Lebensereignisscript vollständig ausgefüllt wurde, wird eine Reihe von Empfehlungen angezeigt. Diese Empfehlungen umfassen Folgendes:

- Kommunale Einrichtungen, die Peter Hilfe und Unterstützung bieten können.
- Staatliche Programme, die in Peters Situation relevant sein könnten, wie beispielsweise Arbeitslosenunterstützung.

Einige Tage nach der Übermittlung des Lebensereignisses meldet sich Peter erneut bei seinem Bürgerkonto an. Er findet eine Nachricht auf seiner Startseite vor. Für Peter wurde ein Leistungsfall erstellt, und aufgrund der Änderungen im Lebensereignis muss die Behörde, die diesen Leistungsfall verwaltet, weitere Informationen zu Peters Einkommenssituation erfassen. Nachdem Peter ein weiteres Fragescript ausgefüllt hat, kehrt er zu den Lebensereignisseiten zurück und überprüft die Informationen in seinem zuvor übermittelten Lebensereignis 'Arbeitsplatz verloren'. Dort findet er die Informationen, die er an die Behörde gesendet hat, und kann sich noch einmal die kommunalen Dienstleistungen ins Gedächtnis rufen, die ihm daraufhin empfohlen worden waren.

Aus Peters Sicht stellt sich die Situation wie folgt dar:

- Er hat einer Behörde oder möglicherweise auch mehreren unterschiedlichen Behörden von seiner misslichen Lage berichtet, ohne die Behörden einzeln kontaktieren zu müssen.
- Ihm wurden Dienstleistungen empfohlen, die in seiner Gemeinde in der Nähe seines Wohnortes angeboten werden. Möglicherweise hat er zuvor nicht einmal gewusst, dass es solche Dienstleistungen überhaupt gibt.
- Ihm wurde geraten, entsprechende staatliche Programme zu beantragen.

Aus Sicht beteiligter Sozialdienste und -behörden stellt sich die Situation wie folgt dar:

- Sie kennen den entsprechenden Kontext. So wissen sie nicht nur, dass Peter ein Programm beantragt hat, sondern auch, was ihn dazu veranlasst hat.
- Das Bürgerkontosystem hat für Peter eine Triage (Sichtung) durchgeführt und dabei wertvolle Ressourcen gespeichert.
- Peter wurde auf kommunale / ehrenamtliche Ressourcen verwiesen, die ihm helfen können.
- Wenn für Peter bereits andere Fälle vorliegen, die mithilfe von Cúram verwaltet werden, können Informationen aus dem Lebensereignis diesen Fällen automatisch zugeführt werden.

Vorgehensweise beim Erstellen eines Lebensereignisses

Analyse

In diesem Abschnitt wird erläutert, wie die Analyse durchgeführt wird, die erforderlich ist, um ein Lebensereignis für Universal Access zu entwerfen. Es ist möglich, Lebensereignisse für Fallbearbeiter zu erstellen oder die Lebensereignisinfrastruktur zu verwenden, um andere Prozesse wie beispielsweise eine Zertifizierung zu steuern; diese Themen sind jedoch nicht Gegenstand des vorliegenden Kapitels. Kenntnisse in Java-Codierung sind keine Voraussetzung für das Entwickeln von Lebensereignissen. In Abhängigkeit von den jeweiligen Anforderungen können viele, wenn nicht gar alle der erforderlichen Artefakte von einem Analysten (Bearbeiter) entwickelt werden. Anhand der Informationen in diesem Abschnitt können Bearbeiter ermitteln, ob für die Implementierung eines Lebensereignisses Java-Entwickler benötigt werden oder nicht.

Im Allgemeinen gibt es zwei Versionen von Lebensereignissen für Bürger:

- Standardlebensereignisse
- Umlauf-Lebensereignisse (Roundtrip-Lebensereignisse)

Standardlebensereignisse ermöglichen es dem Bürger, neue Informationen einzugeben und diese an die Behörde zu übermitteln. Beispiel: Linda meldet sich bei Universal Access an und übermittelt das Lebensereignis 'Schwangerschaft'. Diese Informationen sind gänzlich neu und müssen mit keiner vorhergehenden Situation in Zusammenhang stehen. Sollte sich herausstellen, dass Linda bei der Angabe der Informationen einen Fehler gemacht hat (und beispielsweise nicht den richtigen Namen des Geburtshelfers angegeben hat), kann sie einfach ein neues Lebensereignis starten und alle neuen Informationen vor der Übermittlung erneut eingeben.

Roundtrip-Lebensereignisse sind komplexer. Der Unterschied zwischen diesen Lebensereignissen und Standardlebensereignissen wird daran festgemacht, ob die Daten, die vorab im Lebensereignis ausgefüllt werden, vom Benutzer geändert werden dürfen oder nicht. Wenn vom Bürger erwartet wird, vorab im Ereignis ausgefüllte Informationen zu aktualisieren und nicht nur neue Informationen hinzuzufügen, sollte das betreffende Lebensereignis als Roundtrip-Lebensereignis betrachtet werden. Für diesen Typ Lebensereignis ist es deutlich schwieriger, entsprechende Scripts zu entwerfen.

Ein einfaches Lebensereignis besteht im Wesentlichen aus den folgenden Hauptartefakten:

- Einem IEG-Script mit dem zugehörigen Datenspeicherschema
- Einem IEG-Script zur Prüfung von Antworten in einem zuvor übermittelten Lebensereignis (optional)

- Der Spezifikation einer Cúram-Datenzuordnungsengine, die vorgibt, wie Daten aus dem IEG-Script in den Angaben in den Kundenfällen zugeordnet werden.

Alle diese Artefakte können mithilfe der Administratorbenutzerschnittstelle konfiguriert werden. Weitere Informationen zum Konfigurieren einfacher Lebensereignisse mithilfe der Administratorbenutzerschnittstelle finden Sie im Kapitel zum Thema 'Lebensereignisse konfigurieren' im Handbuch zur Cúram Universal Access-Konfiguration (`Cúram Universal Access Configuration Guide`).

Das Lebensereignissystem kann Informationen, die vom Benutzer eingegeben werden, verarbeiten und die folgenden beiden Aktionen mit diesen Informationen ausführen:

1. Wenn der Benutzer mit dem lokalen Cúram-Fallverarbeitungssystem verknüpft ist, kann das Lebensereignissystem zugehörige Angaben in allen vorhandenen Fällen des betreffenden Benutzers aktualisieren.
2. Wenn der Benutzer mit fernen Systemen verknüpft ist, kann das Lebensereignissystem Aktualisierungen über Web-Services an zugehörige ferne Systeme senden.

Wenn es sich bei dem Lebensereignis um ein Roundtrip-Lebensereignis handelt oder es erforderlich ist, die Angaben der Person in Cúram zu aktualisieren, müssen bestimmte Entwicklungsschritte durchgeführt werden. Im vorliegenden Kapitel werden die Anwendungsprogrammierschnittstellen erläutert, die erforderlich sind, um diese Voraussetzungen zu erfüllen oder um das Verhalten von Standardlebensereignissen durch zusätzliche angepasste Funktionen zu ergänzen.

Erweiterte Lebensereignisse anpassen

Zweck

Das vorliegende Kapitel dient dem folgenden Zweck:

- Beschreibung der Unterschiede zwischen einem erweiterten Lebensereignis und einem einfachen Lebensereignis.
- Beschreibung der Vorgehensweise beim Entwickeln erweiterter Lebensereignisse.

Zielgruppe

Das vorliegende Kapitel richtet sich an Softwarearchitekten und Entwickler.

Übersicht

Das vorliegende Kapitel soll dem Leser eine vollständige Anleitung zur Entwicklung erweiterter Lebensereignisse an die Hand geben.

Abschnitt 1 beschreibt die allgemeine Architektur von erweiterten Lebensereignissen und erörtert, wie die Analyse- und Entwicklungstasks beim Erstellen eines erweiterten Lebensereignisses ausgeführt werden.

Abschnitt 2 beschreibt die Java-Anwendungsprogrammierschnittstelle für erweiterte Lebensereignisse.

Erweiterte Lebensereignisse und deren Einsatzgebiete

Erweiterte Lebensereignisse ermöglichen einen vollständig automatisierten Umlauf (Roundtrip) von Daten. Dies bedeutet, dass Kundenangaben in den Datenspeicher für ein IEG-Script gelesen werden. Anschließend werden diese Daten vom Kunden

aktualisiert. Bei der Übermittlung des Lebensereignisses werden die ursprünglichen Kundenangaben aktualisiert, die in das IEG-Script gelesen wurden. Erweiterte Lebensereignisse werden nur dann benötigt, wenn ein solcher automatisierter Umlauf von Daten erforderlich ist. Unter allen anderen Umständen werden einfache Lebensereignisse empfohlen. Projektarchitekten sollten sorgfältig überlegen, ob ein Umlauf erforderlich ist oder ob die vom Kunden eingegebenen Daten als neue Angaben behandelt werden können, die in die Fälle des Kunden integriert werden sollen.

Erweiterte Lebensereignisse können nicht über die Benutzerschnittstelle der Administration konfiguriert werden, sondern müssen von Entwicklern erstellt werden.

Vorgehensweise beim Erstellen eines Lebensereignisses

Analyse

Der Unterschied zwischen diesen Roundtrip-Lebensereignissen und Standardlebensereignissen wird daran festgemacht, ob die Daten, die vorab im Lebensereignis ausgefüllt werden, vom Benutzer geändert werden dürfen oder nicht. Wenn vom Bürger erwartet wird, vorab im Ereignis ausgefüllte Informationen zu aktualisieren und nicht nur neue Informationen hinzuzufügen, sollte das betreffende Lebensereignis als Roundtrip-Lebensereignis betrachtet werden. Die Entwicklung eines Lebensereignisses dieses Typs ist deutlich schwieriger. Das Subsystem für erweiterte Lebensereignisse wurde speziell für die Anforderungen beim Entwickeln von Roundtrip-Lebensereignissen entwickelt. Das vorliegende Kapitel enthält Informationen dazu, wie ein erweitertes Lebensereignis entwickelt wird, das einen Umlauf (Roundtrip) der Kundeninformationen unterstützt.

Ein erweitertes Lebensereignis besteht im Wesentlichen aus den folgenden Hauptartefakten:

- Einem IEG-Script mit dem zugehörigen Datenspeicherschema
- Einem IEG-Script zur Prüfung von Antworten in einem zuvor übermittelten Lebensereignis (optional)
- Einen Regelwerk für Empfehlungen, das die Gruppe von Empfehlungen auf Basis der im IEG-Script eingegebenen Informationen generiert (optional)

Das Lebensereignissystem kann Informationen, die vom Benutzer eingegeben werden, verarbeiten und die folgenden beiden Aktionen mit diesen Informationen ausführen:

1. Wenn der Benutzer mit dem lokalen Cúram-Fallverarbeitungssystem verknüpft ist, kann das Lebensereignissystem zugehörige Angaben in allen vorhandenen Fällen des betreffenden Benutzers aktualisieren.
2. Wenn der Benutzer mit fernen Systemen verknüpft ist, kann das Lebensereignissystem Aktualisierungen über Web-Services an zugehörige ferne Systeme senden.

Das Lebensereignissystem kann so konfiguriert werden, dass es die Genehmigung des Benutzers einholt, bevor die Informationen in Lebensereignissen an ferne Systeme gesendet werden.

Ein Standardlebensereignis, das nur zum Senden von Informationen an ferne Systeme konfiguriert wird, kann über die Verwaltungsanwendung konfiguriert werden. Weitere Informationen hierzu finden Sie im Handbuch zur Universal Access-Konfiguration (Universal Access Configuration Guide).

Wenn es sich bei dem Lebensereignis um ein Roundtrip-Lebensereignis handelt oder es erforderlich ist, Angaben im lokalen Fallverarbeitungssystem zu aktualisieren, müssen bestimmte Entwicklungsschritte durchgeführt werden, um das Lebensereignis zu konfigurieren. Roundtrip-Lebensereignisse müssen vorab ausgefüllt werden. Derzeit wird das Vorabausfüllen von Lebensereignissen nur für Benutzer unterstützt, die über die Rolle eines Betroffenen mit dem lokalen Cúram-Fallverarbeitungssystem verknüpft sind. Um Informationen aus Fällen lesen und diese Fälle aktualisieren zu können, hängt das Lebensereignissystem von einem Subsystem ab, das als Bürgerdatenhub bezeichnet wird. Im verbleibenden Teil dieses Abschnitts werden die Schritte erörtert, die zum Konfigurieren des Bürgerdatenhubs ausgeführt werden müssen.

Der Lebensereignisbroker verwendet den Datenhub, um die erforderlichen Daten zum Ausfüllen des Lebensereignisses abzurufen. Daher muss der Entwickler den Datenhub so konfigurieren, dass diese Daten extrahiert werden. Darüber hinaus sendet der Lebensereignisbroker die aktualisierten Daten auch über den Datenhub zurück. Der Datenhub muss so konfiguriert werden, dass er dem Broker mitteilt, wie diese aktualisierten Daten zu verwenden sind.

Im Folgenden werden einige der Artefakte aufgeführt, die verwendet werden, um den Bürgerdatenhub für das Lesen von Informationen zu konfigurieren:

- Umsetzung (Transform) - Setzt Daten aus dem Rückstellungsfall (Holding Case) in Datenspeicher-XML um.
- Links zur Angabenfilterung (Filter Evidence Links) - Beim Lesen von Bürgerdaten filtern diese Links nur die Angabenentitäten heraus, die beim Lesen aus dem Rückstellungsfall relevant sind.
- Anzeigeprozessoren (View Processors) - Java-Klassen zum Extrahieren von Nicht-Angabendaten in die Datenspeicher-XML.

Im Folgenden werden einige der Artefakte aufgeführt, die verwendet werden, um den Bürgerdatenhub für das Aktualisieren von Informationen zu konfigurieren:

- Umsetzung (Transform) - Konvertiert die Beschreibung einer Datenspeicher-XML-Differenz wieder in Rückstellungsfallangaben.
- Aktualisierungsprozessoren (Update Processors) - Führen weitere Aktualisierungstasks durch oder aktualisieren Nicht-Angabendaten im Zusammenhang mit dem Bürger.

Überlegungen zur Lebensereignisanalyse: Im Folgenden werden einige Überlegungen angestellt, die sich auf die Komplexität bei der Entwicklung eines bestimmten Lebensereignisses beziehen, das Daten aus einem Angabendatenspeicher oder beteiligtenbezogenen Datenspeicher in Cúram lesen oder in diesen Datenspeicher schreiben muss. Diese Überlegungen dienen der Information bei der Analyse zur Entwicklung von Lebensereignissen und daraus resultierenden Einschätzungen.

1. Handelt es sich bei dem Lebensereignis um ein Standardlebensereignis oder um ein Umlauflebensereignis (Roundtrip)?
2. Welche Informationen müssen im IEG2-Script vorab ausgefüllt werden?
3. Welche Angabendaten werden vom Lebensereignis gelesen?
4. Welche Angabendaten werden vom Lebensereignis aktualisiert?
5. Welche anderen Daten außer Angabendaten werden vom Lebensereignis gelesen bzw. aktualisiert?
6. Wie viele Programme/Falltypen werden vom Lebensereignis beeinflusst?

7. Wenn ein Lebensereignis Angaben mit mehreren Fällen gemeinsam verwendet, verwenden die betreffenden Falltypen dann auch untereinander Daten gemeinsam, indem sie einen Angabenbroker verwenden?
8. Sind einem Lebensereignis Empfehlungen zugeordnet? Ist dies der Fall, beziehen sich diese Empfehlungen auf kommunale Dienstleistungen, staatliche Programme oder beides?

Bei diesen Überlegungen stellt die Handhabung von Entitäten, die sich nicht auf Angaben beziehen, die größte Herausforderung dar. Für alle Lebensereignisse, die auch andere Entitäten als Angabenentitäten aktualisieren, sind Entwickler mit Java-Kenntnissen erforderlich.

Komponenten eines Lebensereignisses erstellen

Übersicht: In diesem Abschnitt wird beschrieben, wie die einzelnen Komponenten eines Lebensereignisses erstellt werden, das den Bürgerdatenhub verwendet. Für diesen Abschnitt des Handbuchs sind keinerlei Kenntnisse im Bereich Java erforderlich.

- Vorgehensweise beim Schreiben von IEG-Scripts für Lebensereignisse, einschließlich Prüfungsscripts
- Vorgehensweise beim Schreiben von Regelwerken für Empfehlungen zu Lebensereignissen
- Vorgehensweise beim Vorabausfüllen eines Lebensereignisscripts mithilfe des Bürgerdatenhubs
- Vorgehensweise beim Verarbeiten von Lebensereignisaktualisierungen mithilfe des Bürgerdatenhubs
- Vorgehensweise beim Zusammensetzen der Komponenten

IEG-Scripts für Lebensereignisse schreiben: Ein IEG-Script für ein Lebensereignis wird im Großen und Ganzen so geschrieben wie jedes andere IEG-Script. Weitere Informationen zum Schreiben von IEG2-Scripts im Allgemeinen finden Sie im Entwicklerhandbuch zum Authoring von Scripts in IEG2 (Developer Guide 'Authoring Scripts in IEG2'). Allerdings gibt es bei Lebensereignisscripts eine Reihe von besonderen Aspekten. Im Wesentlichen hängen diese Aspekte davon ab, ob es sich bei dem Lebensereignis um ein Umlauflebensereignis (Roundtrip-Lebensereignis) oder um ein Standardlebensereignis handelt. Zur Erinnerung: Bei einem Roundtrip-Lebensereignis werden Bürgerdaten in den vom IEG-Script verwendeten Datenspeicher gelesen. Anschließend können diese Daten vom Bürger geändert werden, wenn dieser das Lebensereignisscript Seite für Seite durcharbeitet. Betrachten Sie beispielsweise eine Einzelangabe zum Einkommen, die in das Lebensereignisscript gelesen wird. Der Bürger ändert diese Einkommensinformationen und übermittelt sie anschließend. Der Lebensereignisbroker muss sicherstellen, dass die vom Bürger an den Einkommensdaten vorgenommenen Änderungen auch erkannt werden und korrekt an diejenige Einkommensentität zurückgegeben werden, aus der die Daten ursprünglich gelesen wurden. Der Lebensereignisbroker benötigt eine Möglichkeit, Daten von ihrem Ursprung in der Einkommensentität über das Lebensereignisscript und zurück zu derselben Einkommensentität verfolgen zu können. Um dies umzusetzen, muss der IEG-Scriptentwickler eine Markierung im Datenspeicherschema setzen. Es folgt ein Beispiel für die Definition eines Datenspeichers für Einkommen (Income):

```
<xsd:element name="Income">
       <xsd:complexType>
          <xsd:attribute name="incomeType" type="INCOME_TYPE"
             default=""/>
        <xsd:attribute name="cgissIncomeType"
             type="CGISS_INCOME_TYPE"/>
```

```
          <xsd:attribute name="incomeFrequency"
               type="INCOME_FREQUENCY" default=""/>
         <xsd:attribute name="incomeAmount" type="IEG_MONEY"
10            default="0"/>
          <xsd:attribute name="localID" type="IEG_STRING"/>
        <xsd:complexType>
     </xsd:element>
```

Das Attribut `localID` wird vom Cúram-Lebensereignisbroker verwendet, um die eindeutige Identität der Entität zu verfolgen, aus der die Einkommensdaten abgerufen wurden. Wenn diese Entität vom Benutzer geändert und übermittelt wird, kann der Lebensereignisbroker anhand des Werts für `localID` die korrekte Entität ausfindig machen, die infolge der Änderungen im Lebensereignis aktualisiert werden muss. Es gibt eine Reihe weiterer Sondermarkierungen, die im Schema gesetzt werden können, um bei der Bereitstellung automatischer Aktualisierungen für Cúram-Angabenentitäten zu helfen. Diese Markierungen werden in nachfolgenden Abschnitten erörtert.

Beim Entwickeln eines Scripts für ein Umlauflebensereignis (Roundtrip-Lebensereignis) sollte der Entwickler die Effekte berücksichtigen, die das Vorabausfüllen von Daten auf den Ablauf des Scripts haben kann. Als Beispiel hierfür sind insbesondere die bedingten Cluster anzuführen. Bei Lebensereignisscripts sollten bedingte Cluster vermieden werden, die zu vorab ausgefüllten Daten zugeordnet sind. Solche Cluster kommen in Anliegenscripts häufig vor, funktionieren jedoch nicht gut, wenn der Datenspeicher vorab ausgefüllt worden ist. Betrachten Sie beispielsweise ein Lebensereignis im Zusammenhang mit einem Arbeitsplatzverlust; das boolesche Flag `hasJob` wird für die Entität `Person` verwendet, um anzuzeigen, dass die betreffende Person einen Arbeitsplatz hat. Das IEG-Script stellt dem Benutzer eine Frage: "Hat ein Mitglied Ihres Haushalts einen Arbeitsplatz?" Dadurch wird die Anzeige eines bedingten Clusters gesteuert, der diejenigen Mitglieder des Haushalts identifiziert, die einen Arbeitsplatz haben. Wenn die Daten im Datenspeicher jedoch vorab ausgefüllt sind, besteht die Wahrscheinlichkeit, dass bei einer oder mehreren der Entitäten vom Typ `Person` das Flag `hasJob` bereits auf den Wert 'true' (wahr) gesetzt ist. In der aktuellen Implementierung von IEG2 ist es jedoch nicht möglich, dass die Steuerungsfrage "Hat ein Mitglied Ihres Haushalts einen Arbeitsplatz?" standardmäßig den Wert 'true' annimmt, und zwar selbst dann nicht, wenn der Wert für `hasJob` bei mindestens einem Haushaltsmitglied `true` ist. Aus diesem Grund sollte als allgemeine Regel vermieden werden, Steuerungsfragen für bedingte Cluster wie diesem zu verwenden, wenn die von diesen Clustern gesteuerten Felder vorab ausgefüllt werden.

Prüfungsscripts für Lebensereignisse schreiben

Benutzer, die zuvor ein Lebensereignis übermittelt haben, können dieses erneut aufrufen, um die von ihnen gegebenen Antworten zu überprüfen. IEG-Scripts sind eine ideale Möglichkeit, um diese Art von Informationen in einem einfach lesbaren Format Seite für Seite darzustellen. Allerdings ist ein Script, das zur Datenerfassung geeignet ist, nicht unbedingt auch zur Prüfung zuvor übermittelter Daten geeignet. Zum einen sollten die Felder in einem Prüfungsscript nicht editierbar sein. IEG stellt die Funktion einer 'Übersichtsseite' bereit, mit deren Hilfe Zusammenfassungen von Daten dargestellt werden können, die bereits eingegeben wurden. Übersichtsseiten werden als gute Möglichkeit empfohlen, Prüfungsscripts für Lebensereignisse zu schreiben. Weitere Informationen zum Schreiben von IEG2-Scripts finden Sie im Entwicklerhandbuch zum Authoring von Scripts in IEG2 (Developer Guide 'Authoring Scripts in IEG2'). Wird kein Prüfungsscript bereitgestellt, wird das Fragenscript im schreibgeschützten Modus gestartet, wenn ein Benutzer sein Lebensereignis überprüfen will.

Regelwerke für Empfehlungen zu Lebensereignissen schreiben: Nach der Übermittlung eines Lebensereignisses werden dem Benutzer die kommunalen Dienstleistungen in seiner Gegend angezeigt, die aufgrund des soeben übermittelten Lebensereignisses für geeignet gehalten werden. In derselben Anzeige können auch empfohlene staatliche Programme aufgeführt werden, die der Benutzer per Screening selbst prüfen oder als Anliegen aufnehmen kann. Regelwerke für Empfehlungen zu Lebensereignissen müssen das Regelwerk `TriageInterface` und `AbstractTriageResult` wie folgt erweitern:

```
<RuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
  "../../../../CREOLEInfrastructure/xsd/curam/
    creole/xsd/RuleSet.xsd"
  name="LifeEventRecommendationsRuleSet">
  <Class name="LifeEventRecommendation"
    extends="AbstractTriageResult"
    extendsRuleSet="TriageInterfaceRuleSet">
    ...
  </Class>
  ...
</RuleSet>
```

Im Wesentlichen ähnelt das Schreiben eines Regelwerks für Empfehlungen zu Lebensereignissen dem Schreiben eines Regelwerks für Triage. Informationen hierzu finden Sie im Kapitel zur Triage-Anpassung im vorliegenden Handbuch. Regeln für Empfehlungen zu Lebensereignissen unterscheiden sich jedoch dahingehend, dass sie Entscheidungen treffen können, die darauf basieren, ob eine bestimmte Datenspeicherentität von dem Benutzer geändert wurde, der das Lebensereignisscript ausführt, und - falls dies der Fall ist - welche Art von Änderung vorgenommen wurde. So könnte das Regelwerk beispielsweise eine bestimmte Empfehlung aufgrund der Hinzufügung einer neuen Einkommensentität ausgeben, während bei der Änderung einer vorhandenen Einkommensentität eine andere Empfehlung ausgegeben werden könnte. Das nachstehende Beispiel zeigt, wie Rollenattribute zur Unterstützung von Lebensereignisempfehlungen zu einer Klasse vom Typ `Person` hinzugefügt werden:

```
<Class name="Person" xsi:noNamespaceSchemaLocation=
  "http://www.curamsoftware.com/CreoleRulesSchema.xsd">
  <Attribute name="curamDataStoreUniqueID">
    <type>
      <javaclass name="Long"/>
    </type>
    <derivation>
      <specified/>
    </derivation>
  </Attribute>

  <Attribute name="curamHasChanged">
    <type>
      <javaclass name="Boolean"/>
    </type>
    <derivation>
      <specified/>
    </derivation>
  </Attribute>

  <Attribute name="curamChangeType">
    <type>
      <javaclass name="String"/>
    </type>
    <derivation>
```

```
      <specified/>
    </derivation>
  </Attribute>
</Class>
```

Nach der Übermittlung eines Lebensereignisses initialisiert der Lebensereignisbroker eine Regelsitzung und erstellt Regelobjekte, die den Datenspeicherentitäten für das Lebensereignis entsprechen. Anschließend werden diese Regelobjekte auf Basis des Differenzbefehls, der der betreffenden Datenspeicherentität entspricht, geändert. Betrachten Sie hierzu das vorstehend beschriebene Beispiel der Regelklasse 'Person': Wenn die Personenentität im Datenspeicher infolge der Ausführung des IEG-Scripts des Lebensereignisses geändert wurde, dann gibt das Attribut `curamHasChanged` den Wert `true` (wahr) zurück. Das Attribut `curamChangeType` gibt den Typ der vorgenommenen Änderung zurück:

- `DCMDT10001` - Die Entität wurde vom IEG-Script des Lebensereignisses hinzugefügt.
- `DCMDT10002` - Die Entität wurde vom IEG-Script des Lebensereignisses geändert.
- `DCMDT10003` - Die Entität wurde vom IEG-Script des Lebensereignisses entfernt.

Lebensereignis vorab ausfüllen: In diesem Abschnitt werden die Artefakte beschrieben, die entwickelt werden müssen, um das Script eines Lebensereignisses vorab auszufüllen. Hierbei wird Folgendes erläutert:

- Funktionsweise des Datenhubs zum Lesen von Daten
- Vorgehensweise beim Authoring von Umsetzungen für Leseoperationen (Read Transforms)
- Vorgehensweise beim Verwenden vordefinierter Anzeigeprozessoren (View Processors)

Funktionsweise des Datenhubs zum Lesen von Daten

Der Datenhub stellt eine Möglichkeit dar, um Daten zu Bürgern an vielen unterschiedlichen Positionen zu erfassen und diese Daten als XML-Dokument in einem Datenspeicher zurückzugeben. Bei Verwendung des Datenhubs entfällt die Komplexität beim Lesen von Daten, da nicht mehr festgestellt werden muss, woher die Daten stammen und wie sie an ihrer ursprünglichen Position dargestellt werden. Um beispielsweise ein Lebensereignis vom Typ 'Arbeitsplatzverlust' zu steuern, müssen unter Umständen Informationen zum Einkommen, zur Adresse und zur Beschäftigung einer Person zusammengestellt werden. Diese drei Einzelinformationen werden im zugrunde liegenden System möglicherweise unterschiedlich dargestellt. Es kann sogar sein, dass sie in drei oder mehr Systemen abgelegt sind. Der Aufrufende braucht sich darum jedoch nicht zu kümmern. Der Bürgerdatenhub ermöglicht es seinen Nutzern, diese Einzelinformationen in einer einzigen Operation abzurufen. Operationen dieses Typs werden eindeutig benannt und jeweils als Datenhubkontext bezeichnet. Um das Beispiel 'Arbeitsplatzverlust' zu veranschaulichen wird ein Datenhubkontext namens 'CitizenLostJob' (Bürger hat seinen Arbeitsplatz verloren) definiert, der es ermöglicht, die Informationen zum Einkommen, zur Adresse und zur Beschäftigung in einer einzigen Abfrage abzurufen.

Eine der Quellen, aus denen der Datenhub Informationen beziehen kann, sind die Angaben in Fällen. Dies gilt insbesondere für die Angaben im Rückstellungsfall (Holding Case) eines Bürgers. Der Rückstellungsfall kann den Angabenbroker verwenden, um Daten aus einer Reihe von unterschiedlichen integrierten Fällen oder sogar über Web-Services aus anderen Systemen zusammenzustellen. Der Rückstellungsfall unterscheidet sich in gewisser Hinsicht von anderen Fällen. Für jeden Bürger in einem Cúram-System ist jeweils immer nur ein Rückstellungsfall vorhan-

den. Der Rückstellungsfall hat eine Schnittstelle, die es ermöglicht, alle in ihm enthaltenen Angaben im XML-Format zu extrahieren. Dieses XML-Format ist speziell für die Beschreibung von Angaben optimiert. Da dieses Format speziell für die Beschreibung von Angaben optimiert ist, eignet es sich nicht unbedingt für das Einfügen in einen Datenspeicher. Allerdings ist es relativ einfach, Daten, die in einem XML-Format vorliegen, in ein anderes Format umzusetzen, das dieselben Informationen enthält. Hierfür wird eine Sprache namens XSL Transformation (XSLT) verwendet. Weitere Informationen zu XSLT finden Sie unter 'http://www.w3.org/TR/xslt'. Im nächsten Abschnitt wird veranschaulicht, wie XSLT-Umsetzungen zur Verwendung im Datenhub geschrieben werden.

Authoring von Umsetzungen für Leseoperationen

Um Umsetzungen für den Bürgerdatenhub schreiben zu können, ist es erforderlich, die Struktur der Rückstellungsfall-XML (Quellendaten) und des Datenspeicherschemas (Ziel) zu verstehen. Das Lebensereignis 'CitizenLostJob' ist sehr komplex. Daher wird in diesem Abschnitt als Einführungsbeispiel ein einfaches fiktives Lebensereignis für Bürger beschrieben, die sich ein neues Auto gekauft haben. Dieses Lebensereignis ist dem Datenhubkontext 'CitizenBoughtCar' zugeordnet. In der Realität würde dieses Ereignis nicht als "Lebensereignis" betrachtet werden. Dennoch ist es ein gutes Beispiel für eine Reihe von Grundsätzen, die beim Erstellen eines Umlauflebensereignisses (Roundtrip-Lebensereignis) zu beachten sind. Betrachten Sie zum Zweck dieses Beispiels das folgende Fragment der Rückstellungsfall-XML, mit dem ein Fahrzeug (vehicle) beschrieben wird:

```
<?xml version="1.0" encoding="UTF-8"?>
  <client-data
  xmlns="http://www.curamsoftware.com/schemas/ClientEvidence">
    <client localID="101" isPrimaryParticipant="true">
      <evidence>
        <entity localID="-416020015578349568" type="ET10081">
          <attribute name="vehicleMake">VM2</attribute>
          <attribute name="versionNo">2</attribute>
          <attribute name="startDate">20110301</attribute>
          <attribute name="usageCode">VU1</attribute>
          <attribute name="amountOwed">3,200.00</attribute>
          <attribute name="numberOfDoors">0</attribute>
          <attribute name="evidenceID">
            -5315936410157449216
          </attribute>
          <attribute name="monthlyPayment">0.00</attribute>
          <attribute name="vehicleModel">159</attribute>
          <attribute name="year">2008</attribute>
          <attribute name="equityValue">0.00</attribute>
          <attribute name="endDate">10101</attribute>
          <attribute name="fairMarketValue">17,000.00</attribute>
          <attribute name="curamEffectiveDate">20110301
           </attribute>
        </entity>
      </evidence>
    </client>
  </client-data>
```

Abbildung 2. Rückstellungsfall-XML - Beispiel

Das Element `client` stellt Daten dar, die zum Beteiligten mit der ID der Betroffenenrolle 101 gehören. In den Demonstrationsdaten von Cúram handelt es sich bei diesem Beteiligten um Peter Schmidt. Das Element 'client' enthält eine einzelne Angabenentität vom Typ `ET10081`. In der Cúram Common Evidence-Schicht für allgemeine Angaben ist `ET10081` die Angabentypenkennung für Fahrzeugangaben. Das Attribut `localID` zusammen mit dem Angabentyp identifizieren eindeutig das zu-

grunde liegende Angabenobjekt für das Fahrzeug. Diese Daten müssen der Datenspeicher-XML zugeordnet werden, damit sie zum Ausfüllen eine IEG-Scripts verwendet werden können. Nachstehend sehen Sie, wie die vorstehenden Daten in der Datenspeicher-XML dargestellt werden müssen:

```
<?xml version="1.0" encoding="UTF-8"?>
<Application>
    <Person localID="101" isPrimaryParticipant="true"
      hasVehicle="true">
        <Resource resourcePageCategory="RPC4001"
          localID="-416020015578349568" vehicleMake="VM2"
          versionNo="2" amountOwed="3,200.00" vehicleModel="159"
          year="2008" fairMarketValue="17,000.00"
          curamEffectiveDate="20110301">
            <Descriptor/>
        </Resource>
    </Person>
</Application>
```

Abbildung 3. Datenspeicher-XML - Beispiel

Diese XML-Daten müssen dem Schema entsprechen, das zum Erstellen des IEG-Scripts verwendet wird. Bitte beachten Sie, dass die vorstehende XML einem Schema entspricht, bei dem es sich um ein Superset des Schemas `CitizenPortal.xsd` handelt. Es wird empfohlen, das Schema `CitizenPortal.xsd` als Ausgangspunkt für die in Lebensereignissen von Kunden verwendeten Schemas zu verwenden. Diesen Schemas müssen die Markierungsattribute ('Marker') hinzugefügt werden, die für Lebensereignisse benötigt werden. Diese Markierungsattribute schließen die Verwendung von `localID` ein, wie vorstehend erläutert. Datenspeicherschemas für Entitäten können auch die nachstehenden Sondermarkierungen enthalten, die speziell für die Darstellung von Angaben im Rückstellungsfall entwickelt wurden. Das folgende XSLT-Fragment zeigt, wie Fahrzeugbestandsangaben in eine entsprechende Datenspeicherentität umgesetzt werden:

- curamEffectiveDate - Diese Markierung wird dem Gültigkeitsdatum einer Einzelangabe von Cúram zugeordnet.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:x="http://www.curamsoftware.com/
     schemas/DifferenceCommand"
    xmlns:fn="http://www.w3.org/2006/xpath-functions"
      version="2.0">
    <xsl:output indent="yes"/>

    <xsl:strip-space elements="*"/>

    <xsl:template match="update">
        <xsl:for-each select="./diff[@entityType='Application']">
            <xsl:element name="client-data">
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:for-each>
    </xsl:template>

    <xsl:template match="diff[@entityType='Person']">
        <xsl:element name="client">
            <xsl:attribute name="localID">
                <xsl:value-of select="./@identifier"/>
            </xsl:attribute>
            <xsl:element name="evidence">
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:element>
    </xsl:template>

    <xsl:template match="diff[@entityType='Resource']">
        <xsl:element name="entity">

            <xsl:attribute name="type">ET10081</xsl:attribute>
            <xsl:attribute name="action">
                <xsl:value-of select="./@diffType"/>
            </xsl:attribute>
            <xsl:attribute name="localID">
                <xsl:value-of select="./@identifier"/>
            </xsl:attribute>
            <xsl:for-each select="./attribute">
                <xsl:copy-of select="."/>
            </xsl:for-each>


        </xsl:element>
    </xsl:template>


    <xsl:template match="*">
        <!-- do nothing -->
    </xsl:template>
</xsl:stylesheet>
```

Abbildung 4. XSLT-Umsetzung für Fahrzeugressourcendaten

Der Autor des Lebensereignisses, der diese Umsetzung zu seinem Lebensereignis hinzufügt, kann die in einem beliebigen integrierten Fall aufgezeichneten Fahrzeugangaben in ein Datenspeicherformat umwandeln, das in einem IEG-Script angezeigt werden kann, wobei alle Informationen aus dem Angabendatensatz vorab ausgefüllt werden.

Filter für Angaben definieren

Wenn der Rückstellungsfall aufgerufen wird, um eine XML-Darstellung seiner Angaben zurückzugeben, werden standardmäßig alle Angaben für den betreffenden

Bürger zurückgegeben. Dies könnte eine sehr umfangreiche Abfrage sein, die mehr Informationen zurückgibt als benötigt werden. Der Zweck des Links zum Filtern von Angaben besteht darin, für jeden Datenhubkontext zu definieren, welche Angabentypen relevant sind. Ein Link zum Filtern von Angaben kann definiert werden, indem Einträge zu einer DMX-Datei für diesen Link hinzugefügt werden. Das nachstehende Beispiel zeigt eine solche DMX-Datei für einen Link zum Filtern von Angaben. In dieser Datei werden die Informationen definiert, die für das Lebensereignis 'CitizenBoughtCar' zurückgegeben werden sollen:

```
<?xml version="1.0" encoding="UTF-8"?>
<table name="FILTEREVIDENCELINK">
    <column name="FILTEREVLINKID" type="id" />
    <column name="FILTERNAME" type="text" />
    <column name="EVIDENCETYPECODE" type="text" />
   <row>
        <attribute name="FILTEREVLINKID">
            <value>175</value>
        </attribute>
        <attribute name="FILTERNAME">
            <value>CitizenBoughtCar</value>
        </attribute>
        <attribute name="EVIDENCETYPECODE">
            <value>ET10081</value>
        </attribute>
    </row>
</table>
```

Vordefinierte Anzeigeprozessoren (View Processors) verwenden

Bisher ging es schwerpunktmäßig darum, wie Umsetzungen (Transforms) verwendet werden können, um Angabendaten in Datenspeicher-XML zur Verwendung in einem Lebensereignisscript umzuwandeln. Es gibt jedoch noch weitere wichtige Einzelinformationen, die nicht als Angaben dargestellt werden. Im Allgemeinen muss der Autor eines Lebensereignisses angepassten Java-Code entwickeln, um sämtliche Informationen auszufüllen, die nicht als Angaben dargestellt werden. Mithilfe von Java ist es möglich, *Anzeigeprozessoren* (View Processors) zu entwickeln, mit denen Daten, die keine Angaben sind, extrahiert und in die Datenspeicher-XML umgesetzt werden können. Wenn diese Anzeigeprozessoren dem richtigen Datenhubkontext zugeordnet werden, können sie ihre Informationen zusätzlich zu den Daten von Umsetzungen in den Datenspeicher stellen. Der Lebensereignisbroker wird mit einer Reihe von vordefinierten Anzeigeprozessoren ausgeliefert, mit denen bestimmte, häufig verwendete Nicht-Angabendaten eingefügt werden können.

- Anzeigeprozessor für Haushalt (Household View Processor)
- Anzeigeprozessor für Personenadresse (Person Address View Processor)

Der Anzeigeprozessor für Haushalt findet alle Personen, die mit dem momentan angemeldeten Benutzer in Beziehung stehen, und extrahiert diese Personendaten in den Datenspeicher, zusammen mit Informationen dazu, auf welche Weise diese Personen mit dem angemeldeten Bürger in Beziehung stehen. Diese Informationen basieren auf der CEF-Entität `ConcernRoleRelationship`.

Der Anzeigeprozessor für Personenadresse füllt die wichtigsten Einzeldaten zum angemeldeten Bürger aus, wie beispielsweise den Namen und die Sozialversicherungsnummer. Darüber hinaus extrahiert der Prozessor die Wohnadresse und die Postanschrift des angemeldeten Bürgers. Der Anzeigeprozessor für Haushalt und der Anzeigeprozessor für Personenadresse können zusammen in demselben Lebensereigniskontext verwendet werden. Hierbei muss der Anzeigeprozessor für Personenadresse jedoch nach dem Anzeigeprozessor für Haushalt ausgeführt wer-

den. Der nachstehende Auszug zeigt, wie diese Anzeigeprozessoren konfiguriert werden, um für das Lebensereignis 'CitizenBoughtCar' ausgeführt zu werden:

```
<?xml version="1.0" encoding="UTF-8"?>
  <table name="VIEWPROCESSOR">
    <column name="VIEWPROCESSORID" type="id"/>
    <column name="LOGICALNAME" type="text" />
    <column name="CONTEXT" type="text" />
    <column name="VIEWPROCESSORFACTORY" type="text" />
    <column name="RECORDSTATUS" type="text"/>
    <column name="VERSIONNO" type="number"/>
    <row>
      <attribute name="VIEWPROCESSORID">
        <value>4</value>
      </attribute>
      <attribute name="LOGICALNAME">
        <value>CitizenLostJob0</value>
      </attribute>
      <attribute name="CONTEXT">
        <value>CitizenBoughtCar</value>
      </attribute>
      <attribute name="VIEWPROCESSORFACTORY">
        <value>
        curam.citizen.datahub.internal.impl.
        +HouseholdCustomViewProcessorFactory
        </value>
      </attribute>
      <attribute name="RECORDSTATUS">
        <value>RST1</value>
      </attribute>
      <attribute name="VERSIONNO">
        <value>1</value>
      </attribute>
    </row>
    <row>
       <attribute name="VIEWPROCESSORID">
         <value>5</value>
       </attribute>
       <attribute name="LOGICALNAME">
         <value>CitizenLostJob1</value>
       </attribute>
       <attribute name="CONTEXT">
         <value>CitizenBoughtCar</value>
       </attribute>
       <attribute name="VIEWPROCESSORFACTORY">
          <value>
          curam.citizen.datahub.internal.impl.
          +CustomPersonAddressViewProcessorFactory
          </value>
       </attribute>
       <attribute name="RECORDSTATUS">
         <value>RST1</value>
       </attribute>
       <attribute name="VERSIONNO">
         <value>1</value>
       </attribute>
      </row>
   </table>
```

Bitte beachten Sie die Verwendung des Felds `CONTEXT`. Dieses Feld verknüpft den Anzeigeprozessor (`ViewProcessor`) mit dem Lebensereigniskontext 'CitizenBoughtCar'. Dadurch wird sichergestellt, dass dieser Anzeigeprozessor immer dann aufgerufen wird, wenn der Datenhubkontext 'CitizenBoughtCar' aufgerufen wird. Beachten Sie auch die Verwendung eines logischen Namens (`logicalName`), durch den alle Anzeigeprozessoren eindeutig voneinander unterschieden werden. Die Anzeigeprozessoren für einen Datenhubkontext werden in lexikalischer Reihenfolge aus-

geführt, sodass ein Anzeigeprozessor mit dem logischen Namen "AAA" für den Datenhubkontext "CitizenBoughtCar" vor einem Anzeigeprozessor mit dem logischen Namen "AAB" ausgeführt wird.

Aktualisierungen von Lebensereignissen steuern: In diesem Abschnitt werden die Artefakte beschrieben, die entwickelt werden müssen, um die von einem Lebensereignisscript übermittelten Daten verarbeiten zu können. Folgendes wird erläutert:

- Funktionsweise des Datenhubs zur Aktualisierung von Daten
- Vorgehensweise beim Authoring von Umsetzungen für Aktualisierungsoperationen (Update Transforms)
- Vorgehensweise beim Erstellen neuer Fallbeteiligter von Umsetzungen für Aktualisierungsoperationen (Update Transforms)
- Vorgehensweise beim Konfigurieren von Angabenbrokering für Rückstellungsfall (Holding Case)

Funktionsweise des Datenhubs zur Aktualisierung von Daten

Ebenso wie der Bürgerdatenhub den Datenhubkontext für das Lesen von Daten kennt, enthält er auch Datenhubkontexte für das Aktualisieren von Daten. Lebensereignisse verwenden für Lese- und Aktualisierungsoperationen, die ein und demselben Lebensereignis zugeordnet sind, üblicherweise denselben Datenhubkontextnamen. Somit beschreibt der Kontext 'CitizenBoughtCar' (Bürger hat Auto gekauft) nicht nur eine Gruppe von Artefakten für das Vorabausfüllen eines Scripts für das Lebensereignis 'CitizenBoughtCar', sondern ebenfalls eine Gruppe von Artefakten für die Handhabung von Aktualisierungen der Daten des Bürgers, wenn das Script für das Lebensereignis 'CitizenBoughtCar' vollständig ist.

Eine Aktualisierungsoperation für einen bestimmten Kontext des Bürgerdatenhubs kann dazu führen, dass in einer Einzeltransaktion viele unterschiedliche, individuelle Entitäten aktualisiert werden. Die folgenden Artefakte werden nach einer Scriptübermittlung für den Datenhub bereitgestellt:

- Eine Stammentität für den Datenspeicher
- Ein Differenzbefehl
- Ein Datenhubkontextnamen

Die Stammentität des Datenspeichers ist die Ausgangsebene des Datenspeichers, der über das IEG-Script für Lebensereignisse aktualisiert wurde. Der Differenzbefehl ist eine Entität, die beschreibt, worin sich dieser Datenspeicher von demjenigen unterscheidet, der vor dem Start an das IEG-Script übergeben wurde. Diese Entität beschreibt also, wie der Benutzer die Daten infolge der Ausführung des Lebensereignisscripts geändert hat. Diese Unterschiede werden in drei Basistypen unterteilt:

- Erstellungen: Der Benutzer hat infolge der Ausführung des IEG-Scripts eine Datenspeicherentität erstellt.
- Aktualisierungen: Der Benutzer hat infolge der Ausführung des IEG-Scripts eine Entität aktualisiert.
- Entfernungen: Der Benutzer hat infolge der Ausführung des IEG-Scripts eine Entität entfernt.

Von diesen drei Typen kommen Erstellungen und Aktualisierungen am häufigsten vor. Generell sollte es als unerwünschte Praxis gelten, Benutzern zu erlauben, Elemente in Lebensereignisscripts zu entfernen. Standardlebensereignisse zeichnen sich normalerweise durch eine Reihe von Erstellungen aus, während Roundtrip-

Lebensereignisse gewöhnlich eine Mischung aus Erstellungen und Aktualisierungen sind. Der Differenzbefehl wird vom Lebensereignisbroker automatisch generiert, nachdem ein Lebensereignis übermittelt wurde.

Um eine Aktualisierungsoperation des Datenhubs in automatische Aktualisierungen für Angabenentitäten im Rückstellungsfall umzuwandeln, muss eine Aktualisierungsumsetzung (Update Transform) für den Datenhub angegeben werden. Sofern eine Aktualisierung von anderen Entitäten als Angabenentitäten erforderlich ist, muss ein Aktualisierungsprozessor (Update Processor) entwickelt werden. Für diese Aktualisierungsprozessoren ist die Entwicklung von Java-Code erforderlich.

Umsetzungen für Aktualisierungsoperationen schreiben

Umsetzungen für Aktualisierungsoperationen werden ebenso wie Umsetzungen für Leseoperationen (Read Transforms) mithilfe einer einfachen XSLT-Syntax angegeben. Um Aktualisierungsumsetzungen schreiben zu können, muss sich der Autor sowohl mit dem XML-Eingabeformat als auch mit dem XML-Ausgabeformat für Angaben auskennen. Die folgenden Beispiele basieren auf einem Lebensereignis vom Typ 'CitizenHavingABaby' (Schwangerschaft einer Bürgerin). Mit diesem Lebensereignis kann von einer Bürgerin gemeldet werden, dass sie ein Kind erwartet. Es kann die Anzahl der ungeborenen Kinder eingegeben werden, um beispielsweise darauf hinzuweisen, dass Zwillinge erwartet werden. Darüber hinaus können das erwartete Datum der Niederkunft sowie der Name des Vaters des ungeborenen Kindes eingegeben werden. Bei dem Vater kann es sich um einen vorhandenen Fallbeteiligten oder um jemand völlig anderen handeln. Im letzteren Fall müssen Name, Adresse, Sozialversicherungsnummer etc. des Vaters eingegeben werden. Dieses Lebensereignis ist kein Umlauflebensereignis (Roundtrip), da es hier vielmehr um das Erstellen neuer Angaben geht als um das Aktualisieren bereits vorhandener Angaben. Die Eingabe für eine Aktualisierungsumsetzung ist eine XML-basierte Beschreibung des Differenzbefehls des Datenspeichers. Das nachstehende Beispiel veranschaulicht die Differenzbefehls-XML für das Lebensereignis 'CitizenHavingABaby':

```
<update>
  <diff diffType="NONE" entityType="Application">
    <diff diffType="NONE" entityType="Person" identifier="102">
      <diff diffType="CREATE" entityType="Pregnancy">
        <attribute name="numChildren">1</attribute>
        <attribute name="dueDate">20110528</attribute>
        <attribute name="curamDataStoreUniqueID">385</attribute>
      </diff>
    </diff>
    <diff diffType="UPDATE" entityType="Person" identifier="101">
      <attribute name="isFatherToUnbornChild">true</attribute>
      <attribute name="curamDataStoreUniqueID">399</attribute>
    </diff>
  </diff>
</update>
```

Die Differenzbefehls-XML entspricht Knoten für Knoten der Datenspeicher-XML. Jeder Knoten vom Typ *diff* beschreibt, wie die entsprechende Datenspeicherentität durch die Ausführung des IEG-Scripts geändert wurde. Das Attribut `curamDataStoreUniqueID` gibt an, welche Datenspeicherentität geändert haben. Das Attribut `diffType` gibt die Art der Änderung an, beispielsweise `CREATE` (Erstellung), `UPDATE` (Aktualisierung), `NONE` (Keine) oder `REMOVE` (Entfernung). Bei den aufgelisteten Attributen handelt es sich um diejenigen Attribute, die geändert oder den einzelnen Datenspeicherentitäten hinzugefügt wurden. Im vorstehenden Beispiel wurde eine Schwangerschaft für Linda Schmidt (ID der Betroffenrolle 102) mit einem ungeborenen Kind und dem 28. Mai 2011 als erwartetes Datum der Niederkunft

registriert. Als Vater ist Peter Schmidt (ID der Betroffenenrolle 101) angegeben. Weitere Informationen zur Differenzbefehls-XML finden Sie in dem Schema, das sich im Abschnitt zum Thema 'XML-Schema für Differenzbefehl' befindet. Beim Aktualisieren der XML wird eine Reihe weiterer Attribute und Elemente verwendet, die im Folgenden erläutert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
  <client-data>
    <client localID="102">
      <evidence>
        <entity type="ET10074" action="CREATE" localID="">
          <attribute name="numChildren">1</attribute>
          <attribute name="dueDate">20110528</attribute>
          <entity-data entity-data-type="role">
            <attribute type="LG"/>
            <attribute roleParticipantID="102"/>
            <attribute
              entityRoleIDFieldName="caseParticipantRoleID"/>
          </entity-data>
          <entity-data entity-data-type="role">
            <attribute type="FAT"/>
              <attribute roleParticipantID="101"/>
              <attribute participantType="RL7"/>
              <attribute
                entityRoleIDFieldName="fahCaseParticipantRoleID"/>
            </entity-data>
          <entity type="ET10125" action="CREATE">
            <attribute name="comments"> Unborn child 1</attribute>
            <entity-data entity-data-type="role">
              <attribute type="UNB"/>
              <attribute roleParticipantID="102"/>
              <attribute
                entityRoleIDFieldName="caseParticipantRoleID"/>
            </entity-data>
          </entity>
        </entity>
      </evidence>
    </client>
  </client-data>
```

Abbildung 5. Angaben-XML mit Aktualisierungen

Bitte beachten Sie hier die Verwendung des Attributs `action`, das die Aktion beschreibt, die für die zugrunde liegenden Angaben ausgeführt werden soll. Bei dieser Aktion kann es sich beispielsweise um das Erstellen der Angaben oder um das Aktualisieren vorhandener Angaben handeln. Im nächsten Abschnitt wird die Bedeutung des Elements `<entity-data>` erläutert. Das nachstehende Beispiel veranschaulicht die XSL Transformation, die verwendet wird, um die vorstehende Differenz-XML in die vorstehende Angaben-XML umzuwandeln:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This script plucks out and copies all resource-related -->
<!-- entities from output built by the XMLApplicationBuilder -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:x="http://www.curamsoftware.com/
      schemas/DifferenceCommand"
    xmlns:fn="http://www.w3.org/2006/xpath-functions"
    version="2.0">
    <xsl:output indent="yes"/>
    <xsl:strip-space elements="*"/>
    <xsl:template match="update">
        <xsl:for-each select="./diff[@entityType='Application']">
            <xsl:element name="client-data">
                <xsl:apply-templates/>
            </xsl:element>
```

```
    </xsl:for-each>
</xsl:template>
<xsl:template match="diff[@entityType='Person']">
    <xsl:element name="client">
        <xsl:attribute name="localID">
            <xsl:value-of select="./@identifier"/>
        </xsl:attribute>
        <xsl:element name="evidence">
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:element>
</xsl:template>
<xsl:template match="diff[@entityType='Pregnancy']">
    <xsl:element name="entity">
        <xsl:attribute name="type">ET10074</xsl:attribute>
        <xsl:attribute name="action">
            <xsl:value-of select="./@diffType"/>
        </xsl:attribute>
        <xsl:attribute name="localID">
            <xsl:value-of select="./@identifier"/>
        </xsl:attribute>
        <xsl:for-each select="./attribute">
            <xsl:copy-of select="."/>
        </xsl:for-each>
        <xsl:element name="entity-data">
            <xsl:attribute name="entity-data-type">
              Rolle
            </xsl:attribute>
            <xsl:element name="attribute">
                <xsl:attribute name="type">LG</xsl:attribute>
            </xsl:element>
            <xsl:element name="attribute">
                <xsl:attribute name="roleParticipantID">
                    <xsl:value-of select="../@identifier"/>
                </xsl:attribute>
            </xsl:element>
            <xsl:element name="attribute">
                <xsl:attribute name="entityRoleIDFieldName">
                  caseParticipantRoleID
                </xsl:attribute>
            </xsl:element>
        </xsl:element>
        <xsl:element name="entity-data">
            <xsl:attribute name="entity-data-type">
              Rolle
            </xsl:attribute>
            <xsl:element name="attribute">
                <xsl:attribute name="type">FAT</xsl:attribute>
            </xsl:element>
            <xsl:for-each select=
            "../../diff[@entityType='Person']/attribute[
              @name='isFatherToUnbornChild'
              and ./text()='true']">
                <!-- Copy the participant id if a family -->
                <!-- member is the father -->
                <xsl:element name="attribute">
                  <xsl:attribute name="roleParticipantID">
                      <xsl:value-of select="
                        ../@identifier"/>
                    </xsl:attribute>
                </xsl:element>
            </xsl:for-each>
            <!-- Copy details of absent parent -->
            <xsl:call-template name="absentFather"/>
            <xsl:element name="attribute">
                <xsl:attribute name="entityRoleIDFieldName">
                  fahCaseParticipantRoleID
```

```
                </xsl:attribute>
            </xsl:element>
        </xsl:element>
        <xsl:variable name="numBabies">
            <xsl:value-of select="attribute[
              @name='numChildren'
              ]/text()"/>
        </xsl:variable>
        <xsl:call-template name="unbornChildren">
            <xsl:with-param name="count" select="$numBabies"/>
        </xsl:call-template>
    </xsl:element>
</xsl:template>

<xsl:template name="unbornChildren">
    <xsl:param name="count" select="1"/>
    <xsl:if test="$count > 0">
        <xsl:element name="entity">
            <xsl:attribute name="type">ET10125</xsl:attribute>
            <xsl:attribute name="action">
                <xsl:value-of select="./@diffType"/>
            </xsl:attribute>
            <xsl:element name="attribute">
                <xsl:attribute name="name">
                  comments
                </xsl:attribute>
                Unborn child <xsl:value-of select="$count"/>
            </xsl:element>
            <xsl:element name="entity-data">
                <xsl:attribute name="entity-data-type">
                  Rolle
                </xsl:attribute>
                <xsl:element name="attribute">
                    <xsl:attribute name="type">
                      UNB
                    </xsl:attribute>
                </xsl:element>
                <xsl:element name="attribute">
                    <xsl:attribute name=
                       "roleParticipantID">
                        <xsl:value-of select="
                          ../@identifier"/>
                    </xsl:attribute>
                </xsl:element>
                <xsl:element name="attribute">
                    <xsl:attribute name=
                      "entityRoleIDFieldName">
                      caseParticipantRoleID
                    </xsl:attribute>
                </xsl:element>
            </xsl:element>
        </xsl:element>
        <xsl:call-template name="unbornChildren">
            <xsl:with-param name="count" select="$count - 1"/>
        </xsl:call-template>
    </xsl:if>
</xsl:template>

<xsl:template name="absentFather">
    <xsl:element name="attribute">
        <xsl:attribute name="participantType">
            <xsl:text>RL7</xsl:text>
        </xsl:attribute>
    </xsl:element>

    <xsl:if  test="attribute[@name='fahFirstName']">
        <xsl:element name="attribute">
```

```
                <xsl:attribute name="firstName">
                    <xsl:value-of select="attribute[
                      @name='fahFirstName'
                    ]/text()"/>
                </xsl:attribute>
            </xsl:element>
        </xsl:if>

        <!-- etc. map other personal details such as -->
        <!-- SSN, date of birth -->

        <xsl:if  test="diff[@entityType='ResidentialAddress']">
            <xsl:if  test="diff[
              @entityType='ResidentialAddress']/attribute[
              @name='street1']">
                <xsl:element name="attribute">
                    <xsl:attribute name="street1">
                        <xsl:value-of select=
                        "diff[
                        @entityType='ResidentialAddress']
                          /attribute[
                        @name='street1']/text()"/>
                    </xsl:attribute>
                </xsl:element>
            </xsl:if>
            <!-- etc. map other parts of residential address -->
        </xsl:if>
    </xsl:template>

    <xsl:template match="*">
        <!-- do nothing -->
    </xsl:template>
</xsl:stylesheet>
```

Umsetzungen zum Erstellen neuer Fallbeteiligter schreiben

Leser, die sich mit dem Thema 'Angaben' auskennen, wissen, dass Angabenentitäten häufig auf Dritte verweisen. So verweisen beispielsweise Angaben zu einer Schwangerschaft über die Rolle eines Fallbeteiligten auf den Vater. Der zugehörige Vater kann eine Person oder ein Anwärter sein. Andere Angabentypen wie beispielsweise `Schüler/Student` können auf eine Bildungseinrichtung verweisen, die als Repräsentant einer Fallbeteiligtenrolle eingegeben wird.

Das XML-Schema für Angaben stellt ein generisches Element namens `<entity-data>` bereit, mit dem spezielle Handhabungsanweisungen für den Bürgerdatenhub angegeben werden können. Der Typ der Handhabung hängt vom angegebenen Entitätsdatentyp (`<entity-data-type>`) ab. Cúram stellt einen speziellen Prozessor für den Entitätsdatentyp `role` (Rolle) zur Verfügung. Dieser Entitätsdatenprozessor für Rollen kann verwendet werden, um neue Fallbeteiligtenrollen zu erstellen oder um auf Fallbeteiligtenrollen für vorhandene Fallbeteiligte zu verweisen. In der Angaben-XML-Ausgabe im vorherigen Abschnitt wird das als `type` bezeichnete Attribut zur Angabe des Typs der Fallbeteiligtenrolle verwendet, beispielsweise FAT für Vater (Father) oder UNB für ungeborenes Kind (Unborn Child). Der an dieser Stelle angegebene Wert muss ein Wert aus der Codetabelle `CaseParticipantRoleType` sein. Mit `roleParticipantID` wird die ID der Beteiligtenrolle (ConcernRoleID) eines vorhandenen Beteiligten im System angegeben. Wird diese Angabe gemacht, versucht das System nicht, einen neuen Fallbeteiligten zu erstellen, sondern verwenden einen Fallbeteiligten mit dieser ID erneut. Der Eintrag `entityRoleIDFieldName` ist der Feldname in der entsprechenden Angabenentität. Im Falle der Angabenentität für Schwangerschaft (`Pregnancy`) beispielsweise

lautet der Name dieses Feldes `fahCaseParticipantRoleID`. Sofern ein neuer Beteiligter erstellt werden muss, werden die folgenden Felder vom Datenprozessor für Rollenentitäten unterstützt:

- participantType: Dies ist ein Codetabelleneintrag aus der Codetabelle `ConcernRoleType`. Beispiel: Verwenden Sie RL7 zum Erstellen eines neuen Anwärters.
- firstName
- middleInitial
- lastName
- SSN
- dateOfBirth
- lastName
- lastName
- street1
- city
- state
- zipCode

Andere Entitäten als Angabenentitäten aktualisieren

In den vorherigen Abschnitten wurde veranschaulicht, wie Lebensereignisse konfiguriert werden können, um Aktualisierungen automatisch zu Angabenentitäten in mehreren integrierten Fällen zuzuordnen. Bisweilen müssen für Lebensereignisse andere Entitäten als Angabenentitäten aktualisiert werden, wie beispielsweise Privatadresse, Beschäftigung oder andere kundenspezifische Entitäten außer Angabenentitäten. Üblicherweise werden diese Entitäten für mehrere Fälle gemeinsam verwendet. Ebenso üblich ist es, dass diese Entitäten nicht demselben gesteuerten Lebenszyklus wie Angabenentitäten folgen. Angaben haben viele Vorteile:

- Angaben sind zeitbezogen.
- Angaben sind fallspezifisch, wobei die fallübergreifende gemeinsame Verwendung von Aktualisierungen mithilfe des Angabenbrokers gesteuert wird.
- Fallbearbeiter können Aktualisierungen ablehnen, die aus externen Quellen wie Universal Access stammen.
- Angaben haben einen Zyklus für 'In Bearbeitung/Genehmigung'.
- Angaben können verifiziert werden.

Andere Entitäten als Angabenentitäten bieten keine dieser Vorteile und Sicherheitseinrichtungen. Analysten sollten Entscheidungen bezüglich der Aktualisierung von anderen Entitäten als Angabenentitäten auf Basis von Lebensereignissen mit entsprechender Vorsicht treffen, insbesondere dann, wenn die Änderungen gleichzeitig auf mehrere Fälle angewendet werden können. Es ist möglich, andere Entitäten als Angabenentitäten zu aktualisieren, doch ist hierfür immer angepasster Code erforderlich. Es wird dringend empfohlen, bei der Entwicklung einer solchen Funktionalität entsprechende Sicherheitseinrichtungen zu integrieren, um sicherzustellen, dass mindestens ein Behördenmitarbeiter die betreffenden Änderungen manuell genehmigt, bevor sie auf das System angewendet werden.

Angabenbroker zur Verwendung mit Rückstellungsfall (Holding Case) konfigurieren: Der Rückstellungsfall (Holding Case) an sich ist von geringem Wert. Wie der Name sagt, handelt es sich lediglich um einen Bereich zum Halten von Angaben, bevor diese anderweitig versendet werden. Nachdem Daten im Rückstellungsfall aktualisiert wurden, besteht das Ziel normalerweise darin, diese Aktualisierungen mittels Brokering an integrierte Fälle zu übermitteln, damit Fallbearbeiter die Änderungen

überprüfen und auf die relevanten Fälle anwenden können. Sobald die Daten für die integrierten Fälle akzeptiert worden sind, erkennt Peter die Vorteile der Übermittlung eines Lebensereignisses, da die aktualisierten Daten sich nunmehr positiv auf die ihm gewährten Leistungen auswirken können. Die Brücke zwischen dem Rückstellungsfall und den integrierten Fällen kann nur dann überwunden werden, wenn die entsprechende Angabenbrokerkonfiguration (Evidence Broker) definiert ist. In diesem Abschnitt wird veranschaulicht, wie dies erreicht werden kann. Hintergrundinformationen zum Angabenbroker enthält das Entwicklerhandbuch für den Cúram-Angabenbroker (*Cúram Evidence Broker Developers Guide*).

Gemeinsame Verwendung von Angaben aus Rückstellungsfall konfigurieren

Es folgt ein Beispiel einer Angabenkonfiguration für die gemeinsame Verwendung der Angaben zu einer Schwangerschaft aus dem Rückstellungsfall mit einem integrierten Fall.

```
<?xml version="1.0" encoding="UTF-8"?>
  <table name="EVIDENCEBROKERCONFIG">
    <column name="EVIDENCEBROKERCONFIGID" type="id"/>
    <column name="SOURCETYPE" type="text" />
    <column name="SOURCEID" type="id" />
    <column name="TARGETTYPE" type="text" />
    <column name="TARGETID" type="id"/>
    <column name="SOURCEEVIDENCETYPE" type="text"/>
    <column name="TARGETEVIDENCETYPE" type="text"/>
    <column name="AUTOACCEPTIND" type="bool"/>
    <column name="WEBSERVICESIND" type="bool"/>
    <column name="SHAREDTYPE" type="text"/>
    <column name="RECORDSTATUS" type="text"/>
    <column name="VERSIONNO" type="number"/>
    <row>
      <attribute name="EVIDENCEBROKERCONFIGID">
        <value>10003</value>
      </attribute>
      <attribute name="SOURCETYPE">
        <value>CT10301</value>
      </attribute>
      <attribute name="SOURCEID">
        <value>10330</value>
      </attribute>
      <attribute name="TARGETTYPE">
        <value>CT5</value>
      </attribute>
      <attribute name="TARGETID">
        <value>4</value>
      </attribute>
      <attribute name="SOURCEEVIDENCETYPE">
        <value>ET10000</value>
      </attribute>
      <attribute name="TARGETEVIDENCETYPE">
        <value>ET10074</value>
      </attribute>
      <attribute name="AUTOACCEPTIND">
        <value>0</value>
      </attribute>
      <attribute name="WEBSERVICESIND">
        <value>0</value>
      </attribute>
      <attribute name="SHAREDTYPE">
        <value>SET2002</value>
      </attribute>
      <attribute name="RECORDSTATUS">
        <value>RST1</value>
      </attribute>
      <attribute name="VERSIONNO">
```

```
      <value>1</value>
    </attribute>
  </row>
</table>
```

Wenn Angaben aus dem Rückstellungsfall mit einem anderen integrierten Fall gemeinsam verwendet werden, sollte der Quellentyp *CT10301* lauten und die Quellen-ID auf 10330 gesetzt werden. Der Quellenangabentyp sollte auf `ET10000` gesetzt werden. Dies ist der Code für alle in Rückstellungsfällen gespeicherten Angaben. Angaben dieses Typs werden als `Bestandsangaben` bezeichnet. Der Zielangabentyp lautet in diesem Fall `ET10074`. In Cúram Common Evidence werden hierdurch Schwangerschaftsangaben identifiziert. Der Typ für die gemeinsame Verwendung von Angaben sollte auf `SET2002` gesetzt werden. Dies ist der Code für die gemeinsame Verwendung von nicht identischen Angaben. Bitte beachten Sie, dass `AUTOACCEPTIND` auf 0 gesetzt ist. Es wird dringend empfohlen, diese Einstellung stets beizubehalten, wenn Angaben aus einem Rückstellungsfall mit einem integrierten Fall gemeinsam verwendet werden. Diese Einstellung bedeutet, dass sämtliche eingehenden Änderungen am Rückstellungsfall des Bürgers immer einem Fallbearbeiter zur Prüfung vorgelegt werden. Unter der Voraussetzung, dass der Fallbearbeiter die Änderungen genehmigt, kann der Link 'Eingehende Angaben' auf der Seite 'Angaben für integrierten Fall' verwendet werden, um die Daten aus dem Rückstellungsfall auf die übliche Art und Weise zu synchronisieren.

Um eine Angabenbrokerkonfiguration für eine angepasste Komponente zu definieren, muss eine DMX-Datei erstellt werden, die eine Konfiguration entsprechend dem vorstehenden Beispiel enthält. Beispiel: `%SERVER_DIR%\components\Custom\data\initial\EBROKER_CONFIG.dmx`

Bei der gemeinsamen Verwendung von Bestandsangaben mit einer Standardangabenentität wie beispielsweise einer Schwangerschaft "kopiert" der Angabenbroker die Bestandsangaben mit den Schwangerschaftsdaten in einen neuen Angabendatensatz für die Schwangerschaft im integrierten Zielfall. Zuvor wurde in vorliegendem Handbuch angedeutet, dass es sich bei Bestandsangaben nicht um "standardmäßige" Angaben handelt. Tatsächlich ist es so, dass diese Angaben in einer XML-Darstellung gespeichert werden, sodass der Angabenbroker beim Kopieren der Bestandsangaben in den Zielangabentyp eine Konvertierung der XML-Daten in standardmäßige Angabendaten vornehmen muss. Zur Unterstützung dieses Konvertierungsprozesses müssen Metadaten angegeben werden. Ein Beispiel dieser Metadaten wird nachstehend veranschaulicht:

```
<?xml version="1.0" encoding="UTF-8"?>
<data-hub-config>
  <evidence-config package="curam.holdingcase.evidence">
    <entity name="HoldingEvidence" ev-type-code="ET10000">
      <attribute name="entityStruct">
        curam.citizen.datahub.holdingcase.holdingevidence.struct.
        +HoldingEvidenceDtls
      </attribute>
    </entity>
    <entity name="Pregnancy" ev-type-code="ET10074">
      <attribute name="entityStruct">
        curam.evidence.entity.struct.PregnancyDtls
      </attribute>
      <related-entity>
        <case-participant-role>
          <attribute name="linkAttribute">
            fahCaseParticipantRoleID
          </attribute>
        </case-participant-role>
        <case-participant-role>
          <attribute name="linkAttribute">
```

```
          caseParticipantRoleID
        </attribute>
      </case-participant-role>
    </related-entity>
  </entity>
 </evidence-config>
</data-hub-config>
```

Die Metadaten beschreiben jede Entität, die aus dem Rückstellungsfall in einen integrierten Fall und *umgekehrt* kopiert werden können. Die Metadaten beschreiben auch die dtls-Strukturen, die zum Erstellen der Zielangaben verwendet werden. Darüber hinaus beschreiben die Metadaten, welche der Attribute in den Fallangaben auf Rollen von Fallbeteiligten verweisen. Mithilfe dieser Informationen wird sichergestellt, dass beim Kopieren der Bestandsangaben nicht einfach blind Rollen-IDs von Fallbeteiligten aus den Bestandsangaben kopiert werden, sondern dass jeweils nach der entsprechenden Rollen-ID eines Fallbeteiligten im Zielfall gesucht wird und die ID gegebenenfalls erstellt wird, falls sie nicht vorhanden ist.

Diese Metadaten werden in einer Anwendungsressource (AppResource) gespeichert. Weitere Informationen zu AppResources finden Sie im Cúram-Entwicklerhandbuch für das Authoring von Scripts in IEG2 (Cúram Developer Guide 'Authoring Scripts in IEG2). Der Ressourcenspeicherschlüssel wird durch die Cúram-Umgebungseigenschaft curam.workspaceservices.datahub.metadata angegeben. Standardmäßig nimmt diese Variable den Wert curam.workspaceservices.datahub.metadata an. Dieser Wert verweist auf Standardmetadaten im Datenhub für Bestandsdaten. Anhand der folgenden Schritte können die Standardmetadaten im Datenhub für Bestandsdaten durch eine angepasste Version ersetzt werden, um alle Angabentypen zu unterstützen, die mittels Brokering aus dem Rückstellungsfall an alle integrierten Fälle verteilt werden müssen.

- Kopieren Sie den Inhalt von %SERVER_DIR%\components\WorkspaceServices\data\initial\clob\DataHubMetaData.xml nach %SERVER_DIR%\components\Custom\data\initial\clob\CustomDataHubMetaData.xml.
- Bearbeiten Sie den Inhalt von CustomDataHubMetaData.xml, um alle Angabenentitäten zu beschreiben, die vom Datenhub aktualisiert werden müssen.
- Erstellen Sie eine Datei namens %SERVER_DIR%\components\Custom\data\initial\APP_RESOURCES.dmx. Fügen Sie dieser Datei einen Eintrag wie folgt hinzu:

```
<?xml version="1.0" encoding="UTF-8"?>
  <table name="APPRESOURCE">
    <column name="resourceid" type="id" />
    <column name="localeIdentifier" type="text"/>
    <column name="name" type="text"/>
    <column name="contentType" type="text"/>
    <column name="contentDisposition" type="text"/>
    <column name="content" type="blob"/>
    <column name="internal" type="bool"/>
    <column name="lastWritten" type="timestamp"/>
    <column name="versionNo" type="number"/>
    <row>
      <attribute name="resourceID">
        <value>10700</value>
      </attribute>
      <attribute name="localeIdentifier">
        <value/>
      </attribute>
      <attribute name="name">
        <value>custom.datahub.metadata</value>
      </attribute>
      <attribute name="contentType">
        <value>text/plain</value>
```

```
      </attribute>
      <attribute name="contentDisposition">
        <value>inline</value>
      </attribute>
      <attribute name="content">
        <value>
          ./Custom/data/initial/clob/CustomDataHubMetaData.xml
        </value>
      </attribute>
      <attribute name="internal">
        <value>0</value>
      </attribute>
      <attribute name="lastWritten">
        <value>SYSTIME</value>
      </attribute>
      <attribute name="versionNo">
        <value>1</value>
      </attribute>
    </row>
  </table>
```

- Erstellen Sie die Datei '%SERVER_DIR%\components\Custom\properties\Environment.xml' und fügen Sie ihr einen Eintrag wie folgt hinzu oder hängen Sie diesen Eintrag an die Datei an:

```
<environment>
   <type name="dynamic_properties">
     <section code="WSSVCS"
       name="Workspace Services - Configuration">
       <variable name="curam.workspaceservices.datahub.metadata"
         value="custom.datahub.metadata" onlyin="all"
         type="STRING">
         <comment>
           Identifies an AppResource used to configure DataHub
           meta-data.
         </comment>
       </variable>
     </section>
   </type>
 </environment>
```

Umlauf (Roundtrip) und Konfiguration der gemeinsamen Verwendung von Angaben mit dem Rückstellungsfall

Im vorherigen Abschnitt wurde beschrieben, wie Daten aus dem Rückstellungsfall mit integrierten Fällen gemeinsam verwendet werden. Für Analysten kann auch die Überlegung sinnvoll sein, ob Angaben in die entgegengesetzte Richtung übertragen werden sollten, d. h. aus den integrierten Fällen in den Rückstellungsfall. Wenn die gemeinsame Verwendung von Angaben aus den integrierten Fällen mit dem Rückstellungsfall konfiguriert ist, können Änderungen, die der Fallbearbeiter an ausgewählen Angaben vornimmt, an den Rückstellungsfall zurückgegeben werden. Dies ist von wesentlicher Bedeutung, wenn in Lebensereignissen Daten aus Angabenentitäten in vorhandenen integrierten Fällen vorab ausgefüllt werden müssen. Das nachstehende Beispiel veranschaulicht, wie Schwangerschaftsangaben für die gemeinsame Verwendung mit dem Rückstellungsfall konfiguriert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<table name="EVIDENCEBROKERCONFIG">
  <column name="EVIDENCEBROKERCONFIGID" type="id"/>
  <column name="SOURCETYPE" type="text" />
  <column name="SOURCEID" type="id" />
  <column name="TARGETTYPE" type="text" />
  <column name="TARGETID" type="id"/>
  <column name="SOURCEEVIDENCETYPE" type="text"/>
  <column name="TARGETEVIDENCETYPE" type="text"/>
```

```
  <column name="AUTOACCEPTIND" type="bool"/>
  <column name="WEBSERVICESIND" type="bool"/>
  <column name="SHAREDTYPE" type="text"/>
  <column name="RECORDSTATUS" type="text"/>
  <column name="VERSIONNO" type="number"/>
  <row>
    <attribute name="EVIDENCEBROKERCONFIGID">
      <value>2</value>
    </attribute>
    <attribute name="SOURCETYPE">
      <value>CT5</value>
    </attribute>
    <attribute name="SOURCEID">
      <value>4</value>
    </attribute>
    <attribute name="TARGETTYPE">
      <value>CT10301</value>
    </attribute>
    <attribute name="TARGETID">
      <value>10330</value>
    </attribute>
    <attribute name="SOURCEEVIDENCETYPE">
      <value>ET10074</value>
    </attribute>
    <attribute name="TARGETEVIDENCETYPE">
      <value>ET10000</value>
    </attribute>
    <attribute name="AUTOACCEPTIND">
      <value>1</value>
    </attribute>
    <attribute name="WEBSERVICESIND">
      <value>0</value>
    </attribute>
    <attribute name="SHAREDTYPE">
      <value>SET2002</value>
    </attribute>
    <attribute name="RECORDSTATUS">
      <value>RST1</value>
    </attribute>
    <attribute name="VERSIONNO">
      <value>1</value>
    </attribute>
  </row>
</table>
```

Bitte beachten Sie, dass im Gegensatz zur gemeinsamen Verwendung von Angaben aus dem Rückstellungsfall mit einem integrierten Fall, das Attribut *AUTOACCEPT-IND* hier auf den Wert 1 gesetzt ist. Dies liegt daran, dass der Zielfall ein Rückstellungsfall ist und Rückstellungsfälle vom Konzept her unbeaufsichtigt verarbeitet werden. Es ist nicht vorgesehen, dass Fallbearbeiter gemeinsam verwendete Elemente im Rückstellungsfall überprüfen, da diese Elemente aus einer autoritativen Quelle stammen, d. h. aus dem integrierten Fall.

Zu berücksichtigende Fragen

Mithilfe einer geeigneten Konfiguration ist es möglich, Daten aus dem Rückstellungsfall mit vielen unterschiedlichen integrierten Fällen gemeinsam zu verwenden. Stellen Sie sich Folgendes vor: Die beiden unterschiedlichen integrierten Fälle A und B sind für die gemeinsame Verwendung von Angaben mit Peters Rückstellungsfall H konfiguriert. In den beiden Fälle A und B ist jeweils separat ein Einkommensangabendatensatz für Peter gespeichert. In Peters Rückstellungsfall werden hierzu zwei separate Einkommensdatensätze angezeigt, und soweit es die Fälle A und B betrifft, handelt es sich *tatsächlich* um zwei vollkommen unabhängige

Datensätze: Die Ansicht zu Peters Einkommen in Fall A und die Ansicht zu Peters Einkommen in Fall B. Peter mag dies jedoch nicht sehr sinnvoll erscheinen, schließlich verfügt er nur über ein Einkommen und verwendet ein einziges Portal für die Kommunikation mit dem/den betreffenden Unternehmen im Sozial- und Gesundheitsbereich (SEM). Er mag sich fragen, warum ihm für ein und dasselbe Einkommen zwei Datensätze angezeigt werden. In Fällen wie diesem, bei dem Daten aus einem einzelnen Rückstellungsfall mit mehreren integrierten Fällen gemeinsam verwendet werden, sollte in Betracht gezogen werden, eine weitere Gruppe von Beziehungen für die gemeinsame Verwendung von Daten einzurichten: von A nach B und von B nach A. Diese Fragestellung muss bereits frühzeitig im Projektlebenszyklus angemessen beurteilt werden.

Zusammensetzung aller Bestandteile: In den vorherigen Abschnitten dieses Kapitels wurde erläutert, wie die einzelnen Bestandteile eines Lebensereignisses erstellt werden. Im vorliegenden Abschnitt wird nun beschrieben, wie diese Bestandteile zu einem vollständigen Lebensereignis zusammengesetzt werden. Neue Lebensereignisse können mithilfe der Administrationsseiten für Lebensereignisse konfiguriert werden. Weitere Informationen zur entsprechenden Vorgehensweise finden Sie im Handbuch zu Cúram Universal Access (*Cúram Universal Access Guide*). Die Administrationsseiten ermöglichen es, neue Lebensereignistypen und Lebensereigniskanäle zu erstellen, Rich Text-Beschreibungen hinzuzufügen und die Lebensereignisse zu IEG-Scripts und Regelwerken für Empfehlungen zuzuordnen. Nachdem alle erforderlichen Entitäten in den Administrationsfenstern erstellt worden sind, können die Daten in eine Gruppe von DMX-Dateien extrahiert werden, die wiederum als Basis für die weitere Entwicklung verwendet werden können. Die folgende Gruppe von Befehlen kann für die Extrahierung der relevanten DMX-Dateien verwendet werden:

```
build extractdata -Dtablename=LifeEventType
build extractdata -Dtablename=LifeEventContext
build extractdata -Dtablename=LifeEventCategory
build extractdata -Dtablename=LifeEventCategoryLink
build extractdata -Dtablename=LocalizableText
build extractdata -Dtablename=TextTranslation
```

Die Tabellen 'LocalizableText' und 'TextTranslation' enthalten alle Beschreibungen von Lebensereignissen, werden jedoch auch mit Übersetzungen von Texten gefüllt, die nichts mit Lebensereignissen zu tun haben. Entwickler sollten diese DMX-Dateien prüfen und alle Einträge, die nicht mit den Beschreibungen relevanter Lebensereignisse in Zusammenhang stehen, entfernen, bevor die DMX-Dateien in das Verzeichnis `%SERVER_DIR%\components\Custom\data\initial\` kopiert werden.

Anleitung zur Anwendungsprogrammierschnittstelle für Lebensereignisse

In diesem Abschnitt wird beschrieben, wie die Java-API für Lebensereignisse und der Bürgerdatenhub verwendet werden. Hierbei wird davon ausgegangen, dass der Leser bereits mit dem Material vertraut ist, das bisher in diesem Kapitel vorgestellt wurde.

Ereignis-APIs für Lebensereignisse

Zum Verständnis dieses Abschnitts muss der Leser mit dem Inhalt vertraut sein, der diesem Abschnitt im vorliegenden Kapitel vorausgeht. Der Lebensereignisbroker wird mit Guice-Ereignissen instrumentiert. Entwickler können Listener schreiben, die sich an diese Ereignisse binden lassen. Folgende Ereignisse sind verfügbar:

- `PreCreateLifeEvent`: Dieses Ereignis wird vor dem Start eines Lebensereignisses aufgerufen.

- PostCreateLifeEvent: Dieses Ereignis wird aufgerufen, nachdem das Lebensereignis initialisiert wurde, also nachdem die Datenhubumsetzung und die Anzeigeprozessoren ausgeführt worden sind.
- PreSubmitLifeEvent: Dieses Ereignis wird aufgerufen, nachdem das Lebensereignis übermittelt worden ist, aber bevor die Aktualisierungsprozessoren ausgeführt worden sind.
- PostSubmitLifeEvent : Dieses Ereignis wird aufgerufen, nachdem das Lebensereignis übermittelt worden ist.

Bitte beachten Sie, dass die die Ereignisse vor und nach der Übermittlung des Lebensereignisse (PreSubmitLifeEvent und PostSubmitLifeEvent) aus einem zurückgestellten Prozess heraus ausgeführt werden, sodass SYSTEM als aktueller Benutzer erwartet wird. Ereignisse im Zusammenhang mit Lebensereignissen sollten niemals versuchen, den Inhalt des betreffenden Lebensereignisses zu ändern. Der nachstehende Codeextrakt zeigt, wie die Listenerklasse MyPreCreateListener an eines dieser Lebensereignisse gebunden werden kann:

```
Multibinder<LifeEventEvents.PreCreateLifeEvent>
   preCreateBinder =
    Multibinder.newSetBinder(binder(),
      new TypeLiteral<LifeEventEvents.PreCreateLifeEvent>() { /**/
    });

  preCreateBinder.addBinding().to(MyPreCreateListener.class);
```

Universal Access-Web-Services

Einführung

In diesem Abschnitt werden die Web-Services von IBM Cúram Universal Access beschrieben, und es wird erläutert, wie Peer-Code entwickelt werden kann, um mit diesen Web-Services zu kommunizieren.

In einigen Szenarios wird IBM Cúram Universal Access von Kunden implementiert, um Interaktionen mit deren Kunden über das Internet zu bearbeiten, während für die Verarbeitung von Fällen ein bereits vorhandenes Altsystem genutzt wird. Um diesen Szenarios gerecht zu werden, kann IBM Cúram Universal Access entsprechend konfiguriert werden, um mithilfe von Web-Services mit verschiedenen fernen Systemen zu kommunizieren.

Universal Access unterstützt die folgenden ausgehenden Web-Services:

- Senden eines Antrags auf Leistungen.
- Zurückziehen eines Antrags auf Leistungen.
- Senden eines Lebensereignisses.

Cúram Universal Access unterstützt die folgenden eingehenden Web-Services:

- Erstellen eines Bürgerkontos in Universal Access.
- Verknüpfen eines Benutzers mit einem fernen System. (Hierdurch erhält der Benutzer die Berechtigung zum Senden von Informationen an die betreffenden Systeme sowie zum Empfangen von Informationen von diesen Systemen.)
- Verknüpfung eines Benutzers mit einem fernen System aufheben.
- Empfangen einer Aktualisierung des Status eines übermittelten Antrags.
- Empfangen einer Aktualisierung des Status einer Anforderung zum Zurückziehen eines Antrags.

- Empfangen einer Bürgernachricht (zur Anzeige auf der Startseite eines Bürgerkontos).
- Empfangen von Zahlungsinformationen.
- Empfangen von Kontaktinformationen zu einem Fall.

Überlegungen zur Sicherheit von Web-Services

Universal Access wurde für die Kommunikation mit einer beliebigen Anzahl an fernen Systemen konzipiert. Diese fernen Systeme können auf der Konfigurationsseite für ferne Systeme in der Administrator- und Universal Access Entry Edition Administrator-Anwendung konfiguriert werden.

Ferne Systeme können Web-Services in Universal Access aufrufen und müssen hierfür den Benutzernamen und das zugehörige Kennwort als Berechtigungsnachweise in einem SOAP-Header bereitstellen. Die entsprechende Vorgehensweise wird in Anhang A anhand von Beispielanforderungen für Web-Services erläutert. Es wird dringend empfohlen, jedem fernen System einen unterschiedlichen Benutzernamen und ein unterschiedliches Kennwort zuzuordnen. Der einem fernen System zugeordnete Benutzername wird im Feld für den Quellenbenutzernamen auf der Konfigurationsseite für ferne Systeme festgelegt. Dadurch, dass jedem fernen System ein anderer Benutzername zugeordnet wird, kann Universal Access ordnungsgemäße datenbasierte Sicherheitsprüfungen für eingehende Serviceanforderungen durchführen. Dies verhindert, dass ein fernes System Anforderungen zur Aktualisierung von Daten senden kann, die eigentlich von einem anderen fernen System verwaltet werden.

Service für Antragsverarbeitung

Antrag empfangen

Dieser ausgehende Web-Service wird von Universal Access auf fernen Systemen aufgerufen. Der Service wird verwendet, um einen Antrag auf Leistungen für mindestens ein Sozialprogramm zu übertragen. Die Web Services Description Language (WSDL) zum Beschreiben dieses Service befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\axis\ProcessApplicationService\ProcessApplicationService.wsdl'.

Eine Web-Service-Anforderung dieses Typs enthält die folgenden Informationen:

- intakeApplicationType - Eine ID zur eindeutigen Identifizierung eines Anliegenantragstyps.
- applicationReference – Eine eindeutige Referenz für einen bestimmten Antrag. Hierbei handelt es sich um eine lesbare ID, die den Kunden nach Abschluss eines Antrags angezeigt wird. Beispiel: 512 oder 756. Die Antragsreferenz wird als Argument für andere Web-Services verwendet und sollte vom Empfänger gespeichert werden.
- applicationLocale – Eine Bezeichnung für die bevorzugte Ländereinstellung des Benutzers, der den Antrag eingegeben hat. Beispiel: en_US. Diese Information sollte vom Empfänger gespeichert werden. Ferne Systeme können eine Vielzahl von Informationen an das Konto des Kunden zurücksenden. Einige dieser Informationen müssen vom Absender entsprechend der bevorzugten Ländereinstellung des Kunden lokalisiert werden.
- submittedDateTime – Datum und Uhrzeit der Übermittlung des Antrags. Das Format für Datum/Uhrzeit entspricht dem XML-Schema. Beispiel: 2012-05-29T15:34:49.000+01:00.

- programsAppliedFor – Hier ist eine Liste der Programme enthalten, die im Rahmen dieses Antrags beantragt wurden. Auf jedes dieser Programme wird mit einer eindeutigen Referenz verwiesen. Diese entspricht dem Wert des Referenzfelds, das im Programmabschnitt der Universal Access-Konfiguration definiert wurde. Beispiel:

```
<ns1:programsAppliedFor>
    <ns1:programTypeReference>CashAssistance</ns1:programTypeReference>
    <ns1:programTypeReference>SNAP</ns1:programTypeReference>
</ns1:programsAppliedFor>
```

- applicationData - Enthält eine base64-codierte Darstellung der Anliegendaten. Bei diesen Anliegendaten handelt es sich um die XML-Darstellung des XML-Datenspeichers, der einem Antrag zugeordnet ist.
- applicationSchemaName – Den Namen des Schemas, das zur Erstellung des Datenspeichers für den Antrag verwendet wird.
- senderIdentification – Identifiziert den Absender der Anforderung. Die Identifikation des Absenders besteht aus zwei Komponenten: 1) Kennung des Systems, aus dem die Anforderung stammt, 2) Konto-ID im Citizen-Arbeitsbereich des Benutzers, der die Anforderung erstellt hat. Die zweite Komponente ist optional; in anonym übermittelten Anträgen ist die zweite Komponente nicht enthalten, in Anträgen, die von einem angemeldeten Benutzer übermittelt wurden, jedoch sehr wohl.
- supplementaryInformation – Optional, für zukünftige Verwendung reserviert.

Der Empfänger dieser Informationen muss die Details des Antrags unter Abgleich mit der Absenderidentifikation und der Anliegenantragsreferenz aufzeichnen.

Bei erfolgreicher Ausführung muss die Implementierung dieses Web-Service den booleschen Wert 'true' (wahr) zurückgeben, um anzuzeigen, dass die Anforderung erfolgreich verarbeitet wurde. Tritt bei der Verarbeitung der Anforderung ein Problem auf, muss ein Fehler zurückgegeben werden, der eine Zeichenfolge zur Angabe der Art des Problems enthält. Die Zeichenfolge sollte entsprechend der Ländereinstellung des Universal Access-Servers lokalisiert werden, da sie in den Serverprotokolldateien angezeigt wird.

Anmerkung: Der Empfänger kann mehrere Anträge mit derselben Anliegenantragsreferenz empfangen. Die Anliegenantragsreferenz ist jedoch für einen bestimmten Absender stets eindeutig. Beispiel: System A und B senden eine Anforderung 'receiveApplication()' an System X. Beide Anforderungen enthalten die Antragsreferenz 256. Hierbei ist jedoch zu beachten, dass der Empfänger niemals zwei Anträge von System A mit der Antragsreferenz 256 erhalten sollte.

Zurückziehungsantrag empfangen

Dieser ausgehende Web-Service wird von Universal Access auf fernen Systemen aufgerufen. Dieser Service wird von Kunden verwendet, um einen Antrag zurückzuziehen, den sie zuvor mit dem Antragsempfangsservice übermittelt haben. Die Web Services Description Language (WSDL) zum Beschreiben dieses Service befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\axis\ProcessApplicationService\ProcessApplicationService.wsdl'. Eine Web-Service-Anforderung dieses Typs enthält die folgenden Informationen:

- applicationReference – Eine eindeutige Referenz für den Antrag, der zurückgezogen werden soll. Diese Referenz verweist auf die ID, die mit der Anforderung des Antragsempfangsservice übermittelt wurde.
- programTypeReference – Eine Referenz, die das Programm identifiziert, das zurückgezogen werden soll. Auf jeden dieser Programmtypen wird mit einer ein-

deutigen Referenz verwiesen. Diese entspricht dem Wert des Referenzfelds, das im Programmabschnitt der Universal Access-Konfiguration definiert wurde. Beispiel: CashAssistance.

- requestSubmittedDateTime – Eine Zeitmarke im Datum/Uhrzeit-Format des XML-Schemas, die angibt, wann die Anforderung übermittelt wurde. Beispiel: 2012-05-29T15:34:49.000+01:00
- withdrawalRequestReason – Dieser Wert für die Begründung des Zurückziehungsantrags stammt aus der Codetabelle 'WithdrawalRequestReason'. Folgende Werte sind möglich:
  - WRES1001 – Beschäftigungsverhältnis
  - WRES1002 – Situationsänderung
  - WRES1003 – Irrtümlich hinterlegt
- withdrawalRequestID – Eine ID zur eindeutigen Identifizierung dieses Zurückziehungsantrags von der Universal Access-Absenderinstanz.
- senderIdentification – Identifiziert den Absender der Anforderung. Die Identifikation des Absenders besteht aus zwei Komponenten: 1) Kennung des Systems, aus dem die Anforderung stammt, 2) Konto-ID im Citizen-Arbeitsbereich des Benutzers, der die Anforderung erstellt hat.
- supplementaryInformation – Optional, für zukünftige Verwendung reserviert.

Nach erfolgreicher Verarbeitung wird als Ergebnis eine entsprechende Antwort (receiveWithdrawalRequestResponse) wie folgt erwartet:

```
<receiveWithdrawalRequestResponse>
  <result>true</result>
</receiveWithdrawalRequestResponse>
```

Die Serviceimplementierung sollte einen Fehler zurückgeben, wenn während der Verarbeitung der Anforderung ein Fehler aufgetreten ist. Die Fehlerzeichenfolge sollte entsprechend der Ländereinstellung des Universal Access-Servers lokalisiert werden, da sie in den Serverprotokolldateien angezeigt wird. Es können unter anderem folgende Probleme auftreten:

- Ein Zurückziehungsantrag mit der betreffenden ID wurde von der entsprechenden Universal Access-Instanz bereits gesendet.
- Die Antragsreferenz, auf die verwiesen wird, wird nicht als Antrag erkannt, der zuvor in einem Aufruf des Antragsempfangsservice von derselben Universal Access-Instanz übermittelt wurde.

Der Zurückziehungsantrag wird von der empfangenden Behörde verarbeitet. Im Anschluss daran sollte eine Antwort in Form einer Aktualisierung des Zurückziehungsantrags gesendet werden. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Antragsaktualisierungsservice

Aktualisierung des Antrags für das Anliegenprogramm

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um das Universal Access-System über Änderungen am Status eines Antrags auf Leistungen zu informieren, der zuvor über den Web-Service für Antragsempfang empfangen wurde. Der Status eines Antrags kann in 'Genehmigt', 'Abgelehnt' oder 'Zurückgezogen' übergehen. Wenn ein Antrag abgelehnt wird, kann eine Begründung hierfür in die Web-Servicenachricht eingefügt werden. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei '<Cúram-

Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\UpdateApplication.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Eine Web-Service-Anforderung dieses Typs enthält die folgenden Informationen:

- curamReferenceID – Diese Referenz-ID muss mit dem Element 'applicationReference' für die entsprechende Antragsempfangsanforderung übereinstimmen.
- programApplicationStatus – Folgende Werte sind für den Status des Programmantrags möglich:
  - IPAS1002 – Zurückgezogen
  - IPAS1003 – Genehmigt
  - IPAS1004 – Abgelehnt
- programApplicationDisposedDateTime – Dies ist die formatierte Zeichenfolge für Datum/Uhrzeit im IBM Cúram-Standardformat nach ISO8601: JJJJMMTT HH:MM:SS.
- programApplicationDenialReason – Optional: Wenn der gesendete Status den Wert IPAS1004 aufweist, ist ein Text mit freiem Format für die Begründung der Ablehnung enthalten. Diese Begründung sollte der Codetabelle 'IntakeProgApplDenyReason' von IBM Cúram entnommen werden.

Die Web-Service-Anforderung muss mit einem Cúram-Sicherheitsberechtigungsnachweis gesendet werden. Informationen hierzu enthält Anhang A mit der SOAP-Beispielnachricht. Der Benutzername im Berechtigungsnachweis muss mit dem Quellenbenutzernamen übereinstimmen, der in den Eintrag für das ferne System eingegeben wurde, der dem Peersystem entspricht, das die Anforderung sendet.

Aktualisierung für Zurückziehungsantrag

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um das Universal Access-System über Änderungen am Status eines Zurückziehungsantrags zu informieren, der zuvor über den Web-Service zum Empfang eines Zurückziehungsantrags übermittelt wurde. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\UpdateApplication.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Eine Web-Service-Anforderung dieses Typs enthält die folgenden Informationen:

- curamReferenceID – Diese Referenz-ID muss mit der Zurückziehungsantrags-ID (withdrawalRequestID) in der entsprechenden Nachricht zum Empfang des Zurückziehungsantrags übereinstimmen.
- withdrawalRequestStatus – Dies ist eine Auflistung mit folgenden möglichen Werten:
  - WREQ1002 – Genehmigt
  - WREQ1003 – Abgelehnt
- resolvedDateTime – Eine Zeitmarke im IBM Cúram-Standardformat nach ISO8601: JJJJMMTT HH:MM:SS.
- withdrawalRequestDenialReason – Optional. Im Falle einer Ablehnung des Zurückziehungsantrags ist dies ein Text mit dem Ablehnungsgrund. Der Absender muss diesen Text entsprechend der Ländereinstellung des Kunden lokalisieren, der den Antrag ursprünglich übermittelt hat.

Anhang A enthält eine SOAP-Beispielanforderung für die Operation vom Typ 'Aktualisierung für Zurückziehungsantrag'.

Bei erfolgreicher Ausführung gibt diese Operation ein Dokument zurück, das darüber informiert, dass die Anforderung erfolgreich war. Bei einem Fehlschlag wird ein Fehler ausgelöst. Mögliche Fehlerursachen:

- Die ID des Zurückziehungsantrags entspricht keiner bekannten Zurückziehungsantrags-ID.
- Der Statusübergang des Zurückziehungsantrags ist ungültig.

Lebensereignisservice

Dieser ausgehende Web-Service wird von Universal Access auf fernen Systemen aufgerufen. Die Web Services Description Language (WSDL) zum Beschreiben dieses Service befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\axis\LifeEventService\LifeEvent.wsdl'.

Eine Anforderung für diesen Web-Service enthält die folgenden Felder:

- lifeEventReference – Beschreibt den Typ des Lebensereignisses, beispielsweise 'Adressänderung'.
- senderIdentification – Identifiziert den Absender der Anforderung. Die Identifikation des Absenders besteht aus zwei Komponenten: 1) Kennung des Systems, aus dem die Anforderung stammt, 2) Konto-ID im Citizen-Arbeitsbereich des Benutzers, der die Anforderung erstellt hat.
- lifeEventData - Enthält eine base64-codierte Darstellung der Lebensereignisdaten. Bei diesen Lebensereignisdaten handelt es sich um die XML-Darstellung des XML-Datenspeichers, der einem Lebensereignis zugeordnet ist.
- lifeEventSchemaName – Der Name des Schemas, das zur Erstellung des Datenspeichers für das Lebensereignis verwendet wird.
- submittedDateTime – Datum und Uhrzeit der Übermittlung des Lebensereignisses. Datum und Uhrzeit entsprechen dem XML-Schema. Beispiel: 2012-05-29T15:34:49.000+01:00
- supplementaryInformation – Optional, für zukünftige Verwendung reserviert.

Die Implementierung sollte eine Antwort vom Typ 'lifeEventResponse' mit dem Inhalt 'true' (wahr) zurückgeben, wenn das Lebensereignis erfolgreich verarbeitet wurde. Tritt bei der Verarbeitung des Lebensereignisses hingegen ein Fehler auf, sollte das System einen Fehler entsprechend der WSDL-Spezifikation zurückgeben.

Kontoerstellungsservice

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um ein Citizen-Arbeitsbereichskonto für Benutzer zu erstellen, die zuvor anonym einen Anliegenantrag übermittelt haben. Der Service führt im Wesentlichen zwei diskrete Funktionen aus:

- Er erstellt ein Konto für einen zuvor anonymen Benutzer.
- Er verknüpft dieses Konto mit dem fernen System, das den Web-Service für Kontoerstellung aufruft.

Wenn ein Benutzer des Citizen-Arbeitsbereichs mit einem fernen System 'verknüpft' ist, bedeutet dies, dass der betreffende Benutzer auf einem fernen System registriert ist und dass das ferne System Anforderungen des Citizen-Arbeitsbereichsbenutzers als Anforderungen erkennt, die sich auf einen bestimmten Fall, auf

bestimmte Fälle oder auf eine Einzelperson im fernen System beziehen. Dies hat ernsthafte Auswirkungen auf die Sicherheit des fernen Systems: Das ferne System, das eine Anforderung zur Verknüpfung eines Benutzers oder zur Erstellung eines Kontos für einen Benutzer sendet, muss von der Identität des Kontoeigners überzeugt sein. Das Schema für die Nutzdaten von Web-Service-Anforderung dieses Typs befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\ExternalAccountCreate.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Eine Kontoerstellungsanforderung enthält die folgenden Informationen:

- firstName – Den Vornamen des Kunden.
- middleName – Den zweiten Vornamen des Kunden. Optional.
- surname – Den Nachnamen des Kunden.
- username – Den Benutzernamen für das neu erstellte Konto.
- password – Das Kennwort für das neu erstellte Konto.
- confirmPassword – Die Bestätigung des Kennworts. Diese Bestätigung muss mit dem Kennwort übereinstimmen.
- secretQuestionType – Den Typ der ausgewählten geheimen Frage, um das Konto des Benutzers freizugeben. Die Werte müssen mit Einträgen aus der Codetabelle 'SecretQuestionType' übereinstimmen. Beispiel: SQT1 – Mädchenname der Mutter.
- answer – Eine Antwort auf die geheime Frage. Diese Angabe ist obligatorisch.
- termsAndConditionsAccepted – Einen booleschen Wert, um anzugeben, dass der Kunde die Bedingungen, unter denen das Konto erstellt wird, akzeptiert hat.
- intakeApplicationReference – Einen Verweis auf die eindeutige Antragsreferenz (applicationReference), die als Teil der Anforderung zum Empfang des Antrags übergeben wird. Wird diese Angabe gemacht, wird eine Verknüpfung zwischen dem Antrag und dem neu erstellten Konto erstellt.
- clientIDOnRemoteSystem – Dies ist eine eindeutige Kennung, die verwendet werden kann, um den Benutzer dieses Kontos auf dem fernen System zu identifizieren. Für diese ID gibt es kein vorgeschriebenes Format. So könnte beispielsweise eine Sozialversicherungsnummer verwendet werden. Allerdings muss es anhand dieser Kennung möglich sein, den Kunden auf dem fernen System eindeutig zu identifizieren.
- sourceSystem – Identifiziert das ferne System, das diese Anforderung gesendet hat. Dieser Name muss dem Namen eines fernen Systems entsprechen, das in der Verwaltungsanwendung konfiguriert ist. Weitere Informationen zum Konfigurieren ferner Systeme finden Sie im Kapitel zum Thema 'Ferne Systeme konfigurieren' im Handbuch zu Cúram Universal Access-Konfiguration (`Cúram Universal Access Configuration Guide`).

Ist diese Anforderung erfolgreich, wird die ID des erstellten Citizen-Arbeitsbereichskontos zurückgegeben. Eventuelle Probleme während der Verarbeitung der Anforderung werden über eine Fehlerantwort markiert. Mögliche Probleme:

- Ein Konto ist der Anliegenantragsreferenz bereits zugeordnet.
- Der Benutzername ist bereits vorhanden.
- Der Benutzername und/oder das Kennwort erfüllen nicht die obligatorischen Mindestanforderungen für Kennwortsicherheit, Länge des Benutzernamens etc.

Service für Verknüpfung

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um ein

Citizen-Arbeitsbereichskonto per Link mit einem fernen System zu verknüpfen. Der Abschnitt zum Kontoerstellungsservice enthält eine allgemeine Erläuterung der Auswirkungen, die das Verknüpfen eines Benutzers haben kann. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\ExternalAccountLink.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Diese Web-Service-Anforderung enthält die folgenden Informationen:

- sourceSystem – Den Namen des fernen Systems, das die Anforderung sendet. Dieser Name muss dem Namen eines fernen Systems entsprechen, das im System konfiguriert ist.
- citizenWorkspaceAccountID – Die eindeutige Konto-ID im Citizen-Arbeitsbereichs.
- clientIDOnRemoteSystem – Dies ist eine eindeutige Kennung, die verwendet werden kann, um den Benutzer dieses Kontos auf dem fernen System zu identifizieren. Für diese ID gibt es kein vorgeschriebenes Format. So könnte beispielsweise eine Sozialversicherungsnummer verwendet werden. Allerdings muss es anhand dieser Kennung möglich sein, den Kunden auf dem fernen System eindeutig zu identifizieren.
- createdByUsername – Den Namen des Benutzers auf dem fernen System, der für diese Anforderung verantwortlich ist.

Bei erfolgreicher Ausführung gibt diese Operation ein Dokument zurück, das darüber informiert, dass die Anforderung erfolgreich war. Bei einem Fehlschlag wird ein Fehler ausgelöst. Mögliche Fehlerursachen:

- Die Konto-ID im Citizen-Arbeitsbereich ist ungültig, nicht vorhanden oder einem inaktivierten Konto zugeordnet.
- Das betreffende Konto im Citizen-Arbeitsbereich ist bereits mit dem entsprechenden fernen System verknüpft.

Service für Verknüpfung aufheben

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um die Verknüpfung eines Citizen-Arbeitsbereichskontos mit einem fernen System aufzuheben. Nach Ausführung dieses Service kann der Benutzer des Kontos, dessen Verknüpfung aufgehoben wurde, beispielsweise keine Lebensereignisse mehr an das betreffende ferne System übermitteln. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\ExternalAccountUnlink.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Diese Web-Service-Anforderung enthält die folgenden Informationen:

- sourceSystem – Den Namen des fernen Systems, das die Anforderung sendet.
- citizenWorkspaceAccountID – Die eindeutige ID des Citizen-Arbeitsbereichskontos, dessen Verknüpfung aufgehoben wird.

Bei erfolgreicher Ausführung gibt diese Operation ein Dokument zurück, das darüber informiert, dass die Anforderung erfolgreich war. Bei einem Fehlschlag wird ein Fehler ausgelöst. Mögliche Fehlerursachen:

- Das angegebene Konto ist nicht vorhanden oder nicht aktiv.

- Das angegebene Konto ist nicht mit dem fernen System verknüpft, das die Anforderung sendet.

Bürgernachricht

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Er wird verwendet, um Bürgernachrichten zu senden, die auf der Startseite eines Benutzers angezeigt werden, wenn dieser sich beim Bürgerkonto anmeldet. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\ExternalCitizenMessage.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Diese Web-Service-Anforderung enthält die folgenden Informationen:

- sourceSystem – Den Namen des fernen Systems, das die Anforderung sendet.
- citizenWorkspaceAccountID – Die eindeutige Konto-ID im Citizen-Arbeitsbereich.
- cityIndustryType – Die Angabe des Branchentyps, der der Nachricht zugeordnet ist. Die Werte für dieses Element müssen mit Codes aus der Codetabelle 'CityIndustry' übereinstimmen.
- relatedID – Den Verweis auf die ID einer zugrunde liegenden Entität im fernen System, auf die sich die Nachricht bezieht. Wenn die Nachricht beispielsweise eine Zahlung betrifft, dann wird die ID dieser Zahlung im fernen System durch die zugehörige ID (relatedID) identifiziert.
- externalCitizenMessageType – Den externen Bürgernachrichtentyp aus der Codetabelle 'ExternalCitizenMessageType'.
- messageTitle – Den Titel der Nachricht. Die Lokalisierung dieses Titels entsprechend der Ländereinstellung des Endbenutzers muss auf dem fernen System erfolgen.
- messageBody – Den Nachrichtentext. Die Lokalisierung dieses Texts entsprechend der Ländereinstellung des Endbenutzers muss auf dem fernen System erfolgen.
- effectiveDate – Optional. Das Datum, ab dem die Nachricht gültig ist. Die Nachricht wird erst ab diesem Datum angezeigt. Das Datum muss im Format 'JJJJ-MM-TT' angegeben werden. Wird kein Gültigkeitsdatum angegeben, wird das aktuelle Datum als Gültigkeitsdatum verwendet.
- expiryDate – Das Datum, an dem die Nachricht ablaufen soll. Nach diesem Datum wird die Nachricht dem Benutzer nicht mehr angezeigt. Das Datum muss im Format 'JJJJ-MM-TT' angegeben werden.
- priority – Einen booleschen Wert, der angibt, ob diese Nachricht eine hohe Priorität besitzt.

Einige Nachrichten sind so konzipiert, dass eine neuere Nachricht eine ältere ersetzen kann. Beispiel: Es wird eine Nachricht zu einer Besprechung gesendet. Die Uhrzeit der Besprechung ändert sich, und es wird eine neue Nachricht mit der aktualisierten Uhrzeit der Besprechung gesendet. Dem Kunden werden nicht beide Nachrichten angezeigt. Vielmehr wird die erste Nachricht durch die zweite ersetzt, sodass nur die zweite Nachricht zu sehen ist. Eine externe Nachricht ersetzt automatisch eine andere externe Nachricht, wenn die folgenden Felder denjenigen einer vorhandenen Nachricht entsprechen: 'sourceSystem', 'externalCitizenMessageType' und 'relatedID'.

Zahlungsservice

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um Informationen zu mindestens einer Zahlung zu übermitteln. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\ExternalPayment.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Diese Web-Service-Anforderung kann eine Zahlung oder auch mehrere Zahlungen enthalten. Dies ermöglicht es dem fernen System, Zahlungen in einem Stapel zusammenzufassen und diese Zahlungen aus Gründen der Systemleistung in einer einzigen Anforderung zu senden. Jede Zahlung kann mit einem ganz eigenen Universal Access-Konto in Zusammenhang stehen. Eine einzelne Zahlung kann nach unterschiedlichen Zahlungskomponenten aufgeschlüsselt sein. Eine Zahlungsaufschlüsselung kann eine Zahlungsposition oder auch mehrere Zahlungspositionen enthalten.

Eine einzelne Zahlung enthält die folgenden Informationen:

- paymentID – Zusammen mit dem Quellensystem wird eine Zahlung hierdurch eindeutig identifiziert.
- sourceSystem – Den Namen des fernen Systems, das die Anforderung sendet. Dieser Name muss dem Namen eines fernen Systems entsprechen, das im System konfiguriert ist.
- citizenWorkspaceAccountID – Die eindeutige Konto-ID im Citizen-Arbeitsbereich.
- cityIndustryType – Die Angabe des Branchentyps, der der Zahlung zugeordnet ist. Die Werte für dieses Element müssen mit Codes aus der Codetabelle 'CityIndustry' übereinstimmen. Optional.
- paymentAmount – Den Überschriftenwert für die Zahlung als Ganzes. Diese Zahlung kann optional in eine Reihe von Position aufgegliedert werden.
- currency – Die Währung, in der die Zahlung ausgeführt wurde, enthält Werte aus der Codetabelle 'Currency'. Optional.
- paymentMethod – Die Methode, mit der die Zahlung ausgeführt wurde, enthält Werte aus der Codetabelle 'MethodOfDelivery'.
- paymentStatus – Der Status der Zahlung (beispielsweise 'abgebrochen', 'verarbeitet', 'zurückgestellt' etc.) enthält Werte aus der Codetabelle 'PmtReconciliationStatus'.
- effectiveDate – Das Gültigkeitsdatum der Zahlung im Format JJJJ-MM-TT.
- coverPeriodFrom – Das Startdatum des Zahlungszeitraums für diese Zahlung. Format: JJJJ-MM-TT.
- coverPeriodTo – Das Enddatum des Zahlungszeitraums für diese Zahlung. Format: JJJJ-MM-TT.
- dueDate – Das Datum, zu dem die Zahlung fällig war. Format: JJJJ-MM-TT.
- payeeName – Der Name des Zahlungsempfängers für diese Zahlung.
- payeeAddress – Die Adresse, an die die Zahlung gesendet wurde (im Falle einer Zahlung per Scheck). Optional.
- paymentReferenceNo – Referenz zur eindeutigen Identifizierung der Zahlung in einem bestimmten fernen System.
- bankSortCode - Die Bankleitzahl des Bankkontos, auf das die Zahlung erfolgt.

- bankAccountNo – Die Nummer des Bankkontos, auf das die Zahlung geleistet wird.
- Eine Zahlung kann nach unterschiedlichen Zahlungskomponenten aufgeschlüsselt sein (optional).

Eine Zahlungsaufschlüsselung kann eine Zahlungsposition oder auch mehrere Zahlungspositionen enthalten. Eine Zahlungsposition enthält die folgenden Informationen:

- caseName - Den lesbaren Namen des Falls im fernen System, dem diese Zahlung zugeordnet ist.
- Der Fallname muss entsprechend der Ländereinstellung des Kunden lokalisiert werden. Dieser Fallname muss dem Fallnamen entsprechen, der auf der Seite mit den Kontaktinformationen angezeigt wird.
- caseReference – Referenz zur eindeutigen Identifizierung des Falls in einem bestimmten fernen System.
- componentType – Code für den Komponententyp aus der Codetabelle 'FinComponentType'.
- debitAmount – Der Belastungsbetrag, wenn es sich bei dieser Zahlung um eine Lastschrift handelt.
- creditAmount – Der Habenbetrag, wenn es sich bei dieser Zahlung um eine Gutschrift handelt.
- coverPeriodFrom – Das Startdatum des Zahlungszeitraums für diese Zahlung. Format: JJJJ-MM-TT.
- coverPeriodTo – Das Enddatum des Zahlungszeitraums für diese Zahlung. Format: JJJJ-MM-TT.

Hierbei ist unbedingt zu beachten, dass Zahlungen zuvor übermittelte Zahlungen aufheben können. Beispiel: Eine Zahlung wird über das Testsystem mit der Zahlungs-ID 1234 übermittelt. Anschließend trifft eine weitere Zahlung mit derselben Zahlungs-ID (1234) vom Testsystem ein. Diese Zahlung ersetzt die vorherige Zahlung. Die vorherige Zahlung wird zusammen mit allen zugehörigen Zahlungspositionen physisch entfernt. Ein typisches Beispiel für einen solchen Fall ist der Abbruch einer zuvor ausgegebenen Zahlung.

Kontaktservice

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um ein Register mit Kontaktinformationen von Fallbearbeitern im Zusammenhang mit einem fernen System zu aktualisieren. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei <Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\ExternalContact.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Eine Kontakt-Web-Service-Anforderung enthält die folgenden Informationen:

- sourceSystem – Den Namen des fernen Systems, das die Anforderung sendet. Dieser Name muss dem Namen eines fernen Systems entsprechen, das im System konfiguriert ist.
- contactReference – Eine im fernen Quellensystem eindeutige Referenz für die Kontaktperson.
- fullName – Den vollständigen Namen des Fallbearbeiters.
- phoneNumber – Die Telefonnummer des Fallbearbeiters. Optional.
- mobilePhoneNumber – Die Mobiltelefonnummer des Fallbearbeiters. Optional.

- faxNumber – Die Faxnummer des Fallbearbeiters. Optional.
- email – Die E-Mail-Adresse des Fallbearbeiters. Optional.

Wenn eine Anforderung mit demselben Quellensystem und derselben Kontaktreferenz wie ein bereits vorhandener Eintrag empfangen wird, dann ersetzen die Informationen in der neueren Anforderungen die bereits vorhandenen Informationen.

Fallservice

Hierbei handelt es sich um einen eingehenden Web-Service, der von fernen Systemen in Universal Access aufgerufen wird. Dieser Service wird verwendet, um Informationen von Fällen zu aktualisieren, die einem bestimmten Bürgerkonto zugeordnet sind. Das Schema für die Nutzdaten von Web-Service-Anforderungen dieses Typs befindet sich in der Datei '<Cúram-Verzeichnis>\EJBServer\components\WorkspaceServices\webservices\ExternalCase.xsd'. Eine SOAP-Beispielanforderung für diesen Web-Service ist in Anhang A veröffentlicht.

Eine Web-Service-Anforderung dieses Typs enthält die folgenden Informationen:

- sourceSystem – Den Namen des fernen Systems, das die Anforderung sendet. Dieser Name muss dem Namen eines fernen Systems entsprechen, das im System konfiguriert ist.
- contactReference – Eine im fernen Quellensystem eindeutige Referenz für die Kontaktperson. Diese Referenz muss einer Kontaktpersonreferenz entsprechen, die zuvor über eine Kontaktserviceanforderung übermittelt wurde.
- caseReference – Dies ist eine Fallreferenz, die im fernen System, das die Quelle dieser Anforderung ist, eindeutig sein muss.
- caseName - Den lesbaren Namen des Falls im fernen System. Der Fallname muss entsprechend der Ländereinstellung des Kunden lokalisiert werden. Fallnamen, die im Web-Service für Zahlungen (Payment) verwendet werden, müssen mit Fallnamen übereinstimmen, die in dieser Anforderung angegeben wurden.
- citizenWorkspaceAccountID – Die eindeutige Konto-ID im Citizen-Arbeitsbereich.

Wenn eine Anforderung mit demselben Quellensystem und derselben Fallreferenz wie ein bereits vorhandener Eintrag empfangen wird, dann ersetzen die Informationen in der neueren Anforderungen die bereits vorhandenen Informationen.

Motivationen

Übersicht

In diesem Abschnitt wird die Implementierung von IBM Cúram Universal Access Motivations beschrieben und erläutert, wie eine Motivation implementiert wird.

Anhand von Motivationen können Kunden ihre eigenen Prozesse definieren und diese über das Bürgerportal (Citizen Self-Service) verfügbar machen. Beispiel: Antrag auf Gesundheitsfürsorge. Eine Motivation besteht aus folgenden Elementen:

- IEG-Script zum Erfassen von Kundendaten.
- Datenspeicherschema zum Definieren der Struktur der im IEG-Script erfassten Daten.
- Anzeigeregelwerk zum Definieren der Darstellung der Motivationsergebnisseite.
- Datenregelwerk (optional) zwecks Bereitstellung der Übertragung von Datenspeicherentitäten in Regelobjekten in einem anderen Regelwerk als dem Anzeigeregelwerk.

Das Ergebnis der Ausführung einer Motivation ist eine konfigurierbare Ergebnisseite. Das Script wird verwendet, um eine Gruppe von Fragen zu definieren, die einem Bürger angezeigt werden, wenn dieser eine Motivation initiiert. Die Antworten, die ein Bürger im Script angibt, werden im Datenspeicher abgelegt. Das Anzeigeregelwerk und das Datenregelwerk werden verwendet, um die Ergebnisse auf der Ergebnisseite auszugeben.

Regelwerke

Motivationen definieren ein Anzeigeregelwerk und optional auch ein Datenregelwerk. Das Anzeigeregelwerk steuert die Anzeige der Ergebnisseite, auf der verschiedene Seitenelemente mit dem Inhalt des Ergebnisdatenspeichers in Beziehung stehen. Obwohl ein abstraktes Regelwerk (MotivationRuleSet) bereitgestellt wird, das vom Kunden zu Referenzzwecken verwendet und/oder erweitert werden kann, wird der Vertrag für die Ergebnisseite ausschließlich vom Ergebnisschema gesteuert. Dies bedeutet, dass die Ausgabe der Regeln der Kunden mit dem Ergebnisschema konform sein müssen, und zwar unabhängig davon, ob ein Kunden sich dafür entscheidet, das standardmäßige abstrakte Regelwerk zu Referenzzwecken zu verwenden bzw. zu erweitern oder nicht.

Datenregelwerke

Datenregelwerke werden verwendet, um die allgemeinen Anforderungen für die Konvertierung der im IEG-Datenspeicher erfassten Daten in Regelobjekte zu erfüllen. Die Laufzeit der Motivationen kann die konvertierten Datenspeicherentitäten entweder im Anzeigeregelwerk oder im Datenregelwerk speichern. In bestimmten Situationen ist es pragmatischer, die konvertierten Daten dem Datenregelwerk hinzuzufügen. Dies liegt daran, dass es sich bewährt hat, für jedes Programm ein eigenes Anspruchsberechtigungsregelwerk bereitzuhalten. Das Anzeigeregelwerk hängt zwar vom Anspruchsberechtigungsregelwerk ab, doch ist es nicht ratsam, eine Schleifenabhängigkeit vom Anspruchsberechtigungsregelwerk zurück zum Anzeigeregelwerk zu haben. Aus diesem Grund können die Datenspeicherentitäten in Regelobjekte in einem separaten Datenregelwerk konvertiert werden, und die Anspruchsberechtigungsregelwerke können von diesem Regelwerk abhängig sein, ohne dass es zu irgendwelchen Schleifenabhängigkeiten kommt.

Füllen des Ergebnisdatenspeichers

Das Füllen des Ergebnisdatenspeichers basiert auf der Zuordnung der Ausgabe von den Regeln zum Ergebnisdatenspeicher. Darauf beruht die Anforderung, dass die Ausgabe von den Regeln auf das Schema des Ergebnisdatenspeichers abgestimmt sein muss. Die tatsächliche Zuordnung erfolgt größtenteils automatisch; Kunden können jedoch mithilfe von Anmerkungen Einfluss auf diese Zuordnung nehmen. Das Ergebnisschemaelement (resultSchemaElement) der Anmerkung 'Motivation_Display_Element' ermöglicht die Zuordnung eines von den Regeln ausgegebenen Regelobjekts (RuleObject) zu einem Element im Ergebnisschema, wenn der Name der Regelklasse (RuleClass) und der Name des logisch entsprechenden Elements im Ergebnisschema unterschiedlich sind. Wenn der Name der Regelklasse und der Name des logisch entsprechenden Elements im Ergebnisschema identisch sind, erfolgt die Zuordnung automatisch, und es ist keine Anmerkung erforderlich. Unabhängig davon, ob die Zuordnung von einer Anmerkung gesteuert wird oder nicht, richtet sich die Zuordnung nach dem Inhalt des Ergebnisschemas.

Zuordnung von Regelobjekten

Wenn ein Regelobjekt (RuleObject), d. h. ein Attribut eines Regelobjekts, das wiederum selbst ein Regelobjekt ist, zum Datenspeicher hinzugefügt wird, wird es als neue Datenspeicherentität als untergeordnetes Element der Datenspeicherentität hinzugefügt, die für das übergeordnete Regelobjekt hinzugefügt wurde. Dies erfolgt nur dann, wenn der Name des Regelobjekts im Schema als untergeordnetes Element eines Elements vorhanden ist, bei dem der Name des übergeordneten Elements mit dem Namen des übergeordneten Regelobjekts übereinstimmt. Die Übereinstimmung hängt hier von der Groß-/Kleinschreibung ab; so ergibt sich bei 'person' im Regelobjekt und 'Person' im Schema keine Übereinstimmung. Der Name des Regelobjekts ist das Attribut 'resultSchema' des Elements 'Motivation_Display_Element', sofern eine Anmerkung vorhanden ist. Ist keine Anmerkung vorhanden, ist der Name eines Regelobjekts der Name der Regelklasse (RuleClass). Dies ist wichtig, da dies für den Vergleich mit dem Datenspeicher und die Zuordnung zum Datenspeicher verwendet wird, gegebenenfalls auf der Grundlage dessen, was das Schema zulässt.

Wenn ein einfaches Attribut (ein Attribut eines Regelobjekts, das selbst kein Regelobjekt ist, wie beispielsweise ein Zeichenfolge- oder Codetabellenwert) einem Datenspeicher hinzugefügt wird, wird es als Attribut der Datenspeicherentität hinzugefügt, die für das Regelobjekt hinzugefügt wurde, in dem sich das Attribut befindet. Dies erfolgt nur dann, wenn ein Attribut mit demselben Namen in dem Element in dem Schema vorhanden ist, das dem Entitätstyp des Datenspeichers des Regelobjekts entspricht, das als Eigner fungiert. Der Vergleich von Attributnamen hängt von der Groß-/Kleinschreibung ab; so ergibt sich bei 'dateOfBirth' im Regelobjekt und 'dateOFBIRTH' im Schema keine Übereinstimmung.

Beispielzuordnung von Regelausgabe zu Datenspeicher

Angenommen, es ist ein Eintrag vom Typ 'MotivationType' vorhanden, der das nachstehende Ergebnisschema angibt. Hierbei ist zu beachten, dass nicht alle in diesem Beispiel gezeigten Elemente im abstrakten Regelwerk enthalten oder im Ergebnisdatenspeicher erforderlich sind, sondern teilweise nur zu Demonstrationszwecken aufgeführt werden. Diese Elemente veranschaulichen, wie Kunden vorgehen würden, um angepasste Daten aus ihrem Regelwerk in den Ergebnisdatenspeicher zu füllen.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:d="http://www.curamsoftware.com/BaseDomains" elementFormDefault="qualified">
    <xsd:import namespace="http://www.curamsoftware.com/BaseDomains">
    <xsd:include schemaLocation="IEGDomains">
    <xsd:include schemaLocation="MotivationResultDomains">
    <xsd:element name="Eligibility">
        <xsd:complexType>
            <xsd:sequence minOccurs="0">
                <xsd:element ref="Context" minOccurs="0" maxOccurs="1">
                <xsd:element ref="Results" minOccurs="0" maxOccurs="1">
                <xsd:element ref="ElementNameFromAnnotation" minOccurs="0" maxOccurs="1">
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Context">
        <xsd:complexType>
            <xsd:sequence minOccurs="0">
                <xsd:element ref="Person" minOccurs="0" maxOccurs="unbounded">
                <xsd:element ref="Summary" minOccurs="0" maxOccurs="1">
            </xsd:sequence>
            <xsd:attribute name="extratAttributeNotFromAbstractRuleSet" type="IEG_INT64">
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ElementNameFromAnnotation">
        <xsd:complexType>
            <xsd:attribute name="attributeFromAnnotation" type="IEG_INT64">
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Person">
        <xsd:complexType>
            <xsd:attribute name="personID" type="IEG_INT64">
            <xsd:attribute name="firstName" type="IEG_STRING">
            <xsd:attribute name="lastName" type="IEG_STRING">
            <xsd:attribute name="dateOfBirth" type="IEG_DATE">
            <xsd:attribute name="status" type="CW_MOTIVATION_RESULTS_MEMBER_STATUS">
            <xsd:attribute name="gender" type="IEG_GENDER">
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Summary">
        <xsd:complexType>
            <xsd:attribute name="isRichText" type="IEG_BOOLEAN">
            <xsd:attribute name="summaryText" type="IEG_STRING">
            <xsd:attribute name="title" type="IEG_STRING">
        </xsd:complexType>
    </xsd:element>
```

```
   <xsd:element name="Results">
        <xsd:complexType>
            <xsd:sequence minOccurs="0">
                <xsd:element ref="Category" minOccurs="0" maxOccurs="unbounded">
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Category">
        <xsd:complexType>
            <xsd:sequence minOccurs="0">
                <xsd:element ref="Result" minOccurs="0" maxOccurs="unbounded">
            </xsd:sequence>
            <xsd:attribute name="categoryID" type="IEG_STRING">
            <xsd:attribute name="type" type="IEG_STRING">
            <xsd:attribute name="isPrimary" type="IEG_BOOLEAN">
            <xsd:attribute name="order" type="IEG_INT16">
            <xsd:attribute name="help" type="IEG_STRING">
            <xsd:attribute name="status" type="IEG_STRING">
            <xsd:attribute name="extratAttributeNotFromAbstractRuleSet" type="IEG_INT64">
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Result">
        <xsd:complexType>
            <xsd:sequence minOccurs="0">
                <xsd:element ref="Person" minOccurs="0" maxOccurs="unbounded">
                <xsd:element ref="Benefit" minOccurs="0" maxOccurs="unbounded">
            </xsd:sequence>
            <xsd:attribute name="resultID" type="IEG_STRING">
            <xsd:attribute name="type" type="IEG_CODETABLE_CODE">
            <xsd:attribute name="resultDescription" type="IEG_STRING">
            <xsd:attribute name="status" type="IEG_STRING">
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Benefit">
        <xsd:complexType>
            <xsd:attribute name="benefitType" type="IEG_CODETABLE_CODE">
            <xsd:attribute name="benefitValue" type="IEG_INT32">
            <xsd:attribute name="explanation" type="IEG_STRING">
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Abbildung 6. Datenspeicherschema - Beispiel

Vom Konzept her erlaubt das vorstehende Schema das Füllen des Datenspeichers nur anhand der Regeln wie nachstehend aufgeführt. Jede andere Ausgabe durch die Regeln wird ignoriert.

- Benefit.benefitType
- Benefit.benefitValue
- Benefit.explanation
- Result.Person
- Result.Benefit
- Result.resultID
- Result.type
- Result.resultDescription
- Result.status
- Category.Result
- Category.categoryID
- Category.type
- Category.isPrimary
- Category.order
- Category.help
- Category.status
- Category.extratAttributeNotFromAbstractRuleSet
- Context.Person
- Context.Summary
- Context.extratAttributeNotFromAbstractRuleSet
- Results.Category
- ElementNameFromAnnotation.attributeFromAnnotation
- Eligibility.Context
- Eligibility.Results
- Eligibility.ElementNameFromAnnotation
- Person.personID
- Person.firstName
- Person.lastName
- Person.dateOfBirth
- Person.status
- Person.gender
- Summary.isRichText
- Summary.summaryText
- Summary.title

Beispielregeln: Verarbeitung von Regelobjekten

```
<Class name="Eligibility" extends="AbstractEligibility"
extendsRuleSet="MotivationRuleSet"
xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">
    <Attribute name="context">
       <type>
        <ruleclass name="AbstractContext" ruleset="MotivationRuleSet">
       </type>
         <derivation>
          <create ruleclass="Context">
         </derivation>
    </Attribute>
    <Attribute name="results">
       <type>
        <ruleclass name="AbstractResults" ruleset="MotivationRuleSet">
       </type>
         <derivation>
          <create ruleclass="Results">
         </derivation>
        </Attribute>
    <Attribute name="annotatedAttributeElementWillBeAnnotated">
       <type>
        <ruleclass name="AnnotatedElement">
       </type>
        <derivation>
          <create ruleclass="AnnotatedElement">
        </derivation>
    </Attribute>
</Class>
```

Abbildung 7. Verarbeitung von Regelobjekten - Beispiel

Unter Verwendung der vorstehenden Regelklasse (RuleClass), wobei die Anspruchsberechtigung (Eligibility) die erste Regelklasse im Regelwerk ist (dies ist immer erforderlich), wird bei der Motivationsverarbeitung eine Datenspeicherentität namens 'Eligibility' hinzugefügt, und für jedes Attribut 'Context' (Kontext), 'Results' (Ergebnisse) und 'AnnotatedElement' (Element mit Anmerkungen) werden die zugehörigen Attribute (und die Attribute dieser Attribute etc.) verarbeitet, wobei entsprechend dem jeweiligen Schema neue Datenspeicherentitäten und Attribute zu vorhandenen Entitäten hinzugefügt werden.

Beispielregeln: Komplexe Attribute (Objekt mit einzelner Regel)

```
<Class name="Eligibility" extends="AbstractEligibility"
extendsRuleSet="MotivationRuleSet"
xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">
        <Attribute name="context">
            <type>
                <ruleclass name="AbstractContext" ruleset="MotivationRuleSet">
            </type>
            <derivation>
                <create ruleclass="Context">
            </derivation>
        </Attribute>
        ... Other attributes ..
    </Class>

   <Class name="Context" extends="AbstractContext" extendsRuleSet="MotivationRuleSet"
        xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">
        ... Attributes ...

    </Class>
```

Abbildung 8. Komplexe Attribute

'Context' wurde nicht mit Anmerkungen versehen, sodass der für die entsprechende Datenspeicherentität verwendete Name dem Namen der Regelklasse entspricht, d. h. Context. Bei der Verarbeitung wird das Schema überprüft, um festzustellen, ob 'Eligibility.Context' zulässig ist (d. h. eine Kombination aus dem Namen des übergeordneten Regelobjekts und dem Namen des vorliegenden Regelobjekts). Dies ist aufgrund des Schemas zulässig, sodass eine Entität namens 'Context' der Entität namens 'Eligibility' (Anspruchsberechtigung) hinzugefügt und an diese angehängt wird. Bei Attributen, die auf ein Regelobjekt verweisen, ist der Name des Attributs in der übergeordneten Regelklasse unbedeutend (d. h. in der vorstehenden Regelklasse 'Eligibility' wird der Name des Attributs 'Context' ignoriert). Der Abgleich basiert auf dem Namen des übergeordneten Regelobjekts und dem Namen des Regelobjekts selbst, nicht auf dem Namen des Attributs, das auf das Regelobjekt im übergeordneten Regelobjekt verweist.

Beispielregeln: Komplexe Attribute (Objekt mit einzelner Regel, mit Anmerkungen versehen)

```
<Class name="Eligibility" extends="AbstractEligibility"
extendsRuleSet="MotivationRuleSet"
xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">
         .. Other attributes ..
        <Attribute name="annotatedAttributeElementWillBeAnnotated">
            <type>
                <ruleclass name="AnnotatedElement">
            </type>
            <derivation>
                <create ruleclass="AnnotatedElement">
            </derivation>
        </Attribute>
    </Class>

   <Class name="AnnotatedElement"
        xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">
        <Annotations>
            <Motivation_Display_Element resultSchemaElement="ElementNameFromAnnotation">
        </Annotations>

        .. Attributes ..

    </Class>
```

Abbildung 9. Komplexe Attribute (Objekt mit einzelner Regel, mit Anmerkungen versehen)

'AnnotatedElement' wurde mit Anmerkungen versehen, sodass der für die entsprechende Datenspeicherentität verwendete Name dem Wert des Attributs 'resultSchemaElement' der Anmerkung entspricht, d. h. 'ElementNameFromAnnotation'. Bei der Verarbeitung wird das Schema überprüft, um festzustellen, ob 'Eligibility-.ElementNameFromAnntoation' zulässig ist (d. h. eine Kombination aus dem Namen des übergeordneten Regelobjekts und dem Namen des vorliegenden Regelobjekts). Dies ist aufgrund des Schemas zulässig, sodass eine Entität namens 'ElementNameFromAnnotation' der Entität namens 'Eligibility' (Anspruchsberechtigung) hinzugefügt und an diese angehängt wird. Bei Attributen, die auf ein Regelobjekt verweisen, ist der Name des Attributs in der übergeordneten Regelklasse unbedeutend (d. h. in der vorstehenden Regelklasse 'Eligibility' wird der Name des Attributs 'annotatedAttributeElementWillBeAnnotated' ignoriert). Der Abgleich basiert auf dem Namen des übergeordneten Regelobjekts und dem Namen des Regelobjekts selbst, nicht auf dem Namen des Attributs, das auf das Regelobjekt im übergeordneten Regelobjekt verweist. Hierbei ist Folgendes zu beachten: Wenn die Anmerkung nicht vorhanden wäre, würde der Name 'AnnotatedElement' für die potenzielle Datenspeicherentität verwendet werden. Das Schema lässt 'Eligibility-.AnnotatedElement' nicht zu, sodass diese Entität daher nicht erstellt würde. Dies zeigt, dass eine Regelklasse, die im Ergebnisdatenspeicher vorhanden sein soll, mit Anmerkungen versehen werden muss, um mit den erwarteten Elementen im Schema übereinzustimmen.

Beispielregeln: Komplexe Attribute (Liste von Regelobjekten)

```
<Class name="Context" extends="AbstractContext" extendsRuleSet="MotivationRuleSet"
      xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">
      <Attribute name="householdMembers">
         <type>
              <javaclass name="List">
                  <ruleclass name="AbstractPerson" ruleset="MotivationRuleSet">
              </javaclass>
          </type>
          <derivation>
              <readall ruleclass="Person">
          </derivation>
      </Attribute>
      .. Other attributes ..
  </Class>

 <Class name="Person" extends="AbstractPerson" extendsRuleSet="MotivationRuleSet"
 xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">
    .. Attributes ..
  </Class>
```

Abbildung 10. Komplexe Attribute (Liste von Regelobjekten)

Das vorstehende Attribut 'householdMembers' ist ein komplexes Attribut, das eine Liste von Regelobjekten zurückgibt. Jedes der Regelobjekte in der Liste führt zur Erstellung einer neuen Datenspeicherentität, die an die Datenspeicherentität 'Context' angehängt wird, sofern die Namen dieser Regelobjekte für das Schema geeignet sind. In diesem Fall wird das Schema bei der Verarbeitung überprüft, um festzustellen, ob 'Context.Person' zulässig ist (d. h. eine Kombination aus dem Namen des übergeordneten Regelobjekts und dem Namen des vorliegenden Regelobjekts). Dies ist aufgrund des Schemas zulässig, sodass eine Reihe von Entitäten namens 'Person' der Entität 'Context' hinzugefügt und an diese angehängt wird. Bei Attributen, die auf eine Liste von Regelobjekten verweisen, ist der Name des Attributs in der übergeordneten Regelklasse unbedeutend (d. h. in der vorstehenden Regelklasse 'Context' wird der Name des Attributs 'householdMembers' ignoriert). Der Abgleich basiert auf dem Namen des übergeordneten Regelobjekts und dem Namen eines jeden Regelobjekts selbst, nicht auf dem Namen des Attributs, das auf das jeweilige Regelobjekt im übergeordneten Regelobjekt verweist.

Beispielregeln: Einfache Attribute

```
<Class name="Person" extends="AbstractPerson" extendsRuleSet="MotivationRuleSet"
        xsi:noNamespaceSchemaLocation="http://www.curamsoftware.com/CreoleRulesSchema.xsd">

        <Attribute name="personID">
            <type>
                <javaclass name="Long">
            </type>
            <derivation>
                <specified>
            </derivation>
        </Attribute>

        <Attribute name="medicaidCategory">
            <type>
                <codetableentry table="MotivationTestCategory">
            </type>
            <derivation>
                <specified>
            </derivation>
        </Attribute>

    </Class>
```

Abbildung 11. Einfache Attribute

Bei den vorstehenden Attributen 'personID' und 'medicaidCategory' handelt es sich um einfache Attribute, d. h. es sind keine Regelobjekte. Diese Attribute werden nicht als untergeordnete Entitäten im Datenspeicher hinzugefügt, sondern als Attribute der Datenspeicherentität, die für das jeweils zugehörige übergeordnete Regelobjekt erstellt wurde, sofern die Namen der Attribute für das Schema geeignet sind. Im Gegensatz zu Attributen, die Regelobjekte sind, bei denen der Name des Attributs im übergeordneten Regelobjekt unbedeutend ist, ist der Attributname bei einfachen Attributen hingegen sehr wohl von Bedeutung. Bei der Verarbeitung wird überprüft, ob 'Person.personID' und 'Person.medicaidCategory' für das Schema geeignet sind. 'Person.personID' ist geeignet, sodass 'personID' als Attribut der Entität 'Person' hinzugefügt wird. 'Person.medicaidCategory' ist im Schema jedoch nicht enthalten, sodass 'medicaidCategory' nicht der Entität 'Person' im Datenspeicher hinzugefügt wird.

Vollständig anpassbare Universal Access-Artefakte

Einführung

In diesem Kapitel werden die Artefakte beschrieben, die in Universal Access vollständig angepasst werden können. Darüber hinaus wird erläutert, wie entsprechende Anpassungen an diesen Artefakten vorgenommen werden.

Anpassbarer Universal Access-Seiteninhalt

Öffentliche Universal Access-Seiten werden von XML-Seiten für Seitenwiedergabe gesteuert, die im Anwendungsressourcenspeicher bereitgestellt werden. Jede Seite verfügt über eine entsprechende Eigenschaftendatei und Bilder, die bei der Wiedergabe der betreffenden Seite verwendet werden. Diese Eigenschaftendateien und Bilder befinden sich ebenfalls im Anwendungsressourcenspeicher. Um zum Anwendungsressourcenspeicher zu navigieren, müssen Sie sich bei der Verwaltungsanwendung anmelden, zu Universal Access wechseln, die Option 'Anwendungsressource' auswählen und Ihre Suche in der Anzeige filtern. Alle Texte, Onlinehilfen und Bilder für Seiteninhalte sind anpassbar und lokalisierbar.

Texte und Onlinehilfen

Anfängliche Daten für die auf Universal Access-Seiten verwendete Texte und Onlinehilfen befinden sich in folgendem Verzeichnis:

- *Curam-Verzeichnis\EJBServer\components\CitizenWorkspace\Data_Manager\Initial_Data\blob\prop*

Universal Access verwendet den Mechanismus des Anwendungsressourcenspeichers, um Onlinehilfen, Bilder und Seitentexte für die eigenen Seiten zu konfigurieren. Hilfetexte beispielsweise können jeder Seite in der Universal Access-Anwendung zugeordnet werden. Die Hilfe wird in einer verdeckten Anzeige oben auf der Seite angezeigt. Auf diese Anzeige kann der Benutzer über den Link 'Hilfe' zugreifen.

Jede Universal Access-Seite verfügt über eine entsprechende Anwendungsressource vom Typ 'property', die im Lieferumfang enthalten ist. Um die Onlinehilfe für eine der Universal Access-Seiten zu ändern, muss der Entwickler den Namen der betreffenden Seite und die zugehörige Eigenschaftendatei kennen. Beispiel: Die Seite 'ScreeningOptionalLogin' (d. h. die Seite 'Erste Schritte', die nach Auswahl von 'Bin ich anspruchsberechtigt?' auf der Startseite des Bürgerportals angezeigt wird). Die zugehörige Eigenschaftendatei befindet sich in folgendem Verzeichnis:

- *Curam-Verzeichnis\EJBServer\components\CitizenWorkspace\Data_Manager\Initial_Data\blob\prop\ScreeningOptionalLogin.properties*

Ein Verweis hierauf erfolgt über die folgende DMX-Datei:

- *Curam-Verzeichnis\EJBServer\components\CitizenWorkspace\Data_Manager\Initial_Data\APPRESOURCE_PROP.dmx*

Da eine Änderung an den ursprünglichen DMX-Daten vorgenommen wird, entspricht die Vorgehensweise den empfohlenen Schritten, die zum Ändern aller DMX-Daten gelten. Diese Schritte werden im Handbuch für Cúram-Serverentwickler (`Cúram Server Developer's Guide`) beschrieben.

Bearbeiten Sie einfach Ihre Version der Datei 'ScreeningOptionalLogin.properties' und ändern Sie den Eigenschaftentext entsprechend Ihren Anforderungen. Alle Texte, die von XML-Eigenschaften für Seitenwiedergabe gesteuert werden, können in ähnlicher Weise geändert werden.

Bilder

Ursprüngliche Daten für die auf den Universal Access-Seiten verwendeten Bilder befinden sich in folgendem Verzeichnis:

- *Curam-Verzeichnis\EJBServer\components\CitizenWorkspace\Data_Manager\Initial_Data\blob\img*

Die Vorgehensweise zum Ersetzen von Symbolen/Bilder entspricht der Vorgehensweise zum Ersetzen von Texten. Betrachten Sie beispielsweise die Seite 'ScreeningOptionalLogin'. Die XML-Quellendatei der Seite befindet sich in folgendem Verzeichnis:

- *Data_Manager\Initial_Data\blob\xml\ScreeningOptionalLogin*

Bitte beachten Sie, dass es sich bei dem Symbol, das der Seitenüberschrift zugeordnet ist, um das Symbol 'title_getting_started' handelt. Diese Datei befindet sich in folgendem Verzeichnis:

- *Data_Manager\Initial_Data\blob\img*

Um dieses Bild durch eine anderes zu ersetzen, führen Sie die gleichen Schritte aus, die vorstehend für das Ersetzen von Seitentext bzw. Hilfetext beschrieben sind.

Übersetzung

Es besteht die Möglichkeit, unterschiedliche Eigenschaftendateien zu verwenden, um eine Übersetzung des Seiteninhalts in verschiedene Sprachen bereitzustellen. Das folgende Beispiel zeigt, wie für die Seite 'ScreeningOptionalLogin' eine neue Übersetzung hinzugefügt wird. Im Wesentlichen folgt dieses Beispiel den Richtlinien für das Hinzufügen neuer Einträge zu DMX-Dateien. Diese Richtlinien werden im Handbuch für Cúram-Serverentwickler (`Cúram Server Developer's Guide`) beschrieben.

Um beispielsweise eine Übersetzung der Seite ScreeningOptionalLogin' ins Französische bereitzustellen, erstellen Sie eine neue DMX-Datei:

- *Cúram-Verzeichnis\EJBServer\components\custom\Data_Manager\Initial_Data\APPRESOURCE_PROP.dmx.*

Fügen Sie dieser Datei eine Zeile hinzu, die auf eine neue Datei namens 'blob\prop\ScreeningOptionalLogin_fr.properties' verweist. Der Ressourcenname muss mit dem Ressourcennamen für die englische Version der Eigenschaften übereinstimmen, d. h. ScreeningOptionalLogin. Allerdings enthält die Spalte 'localeIdentifier' für die lokale Kennung in diesem Fall den Wert <value>fr</value>.

Fügen Sie einen neuen Eintrag zu 'project\properties\datamanager_config.xml' hinzu, der auf Folgendes verweist:

- *Cúram-Verzeichnis\EJBServer\components\custom\Data_Manager\Initial_Data\APPRESOURCE_PROP.dmx*

Erstellen Sie die Datei 'Cúram-Verzeichnis\EJBServer\components\custom\Data_Manager\Initial_Data \blob\prop\ScreeningOptionalLogin_fr.properties' und geben Sie für alle relevanten Eigenschaftswerte die entsprechende französische Übersetzung ein.

Bitte lesen Sie im Kapitel zur Anpassung des Bürgerkontos die Informationen zum Hinzufügen neuer Sprachen zu Bürgerkontoseiten.

Universal Access-Seitenwiedergabe - Darstellung und Funktionsweise

Die Darstellung und Funktionsweise von Universal Access kann (in gewissem Umfang) geändert werden, indem die zugehörigen Darstellungseigenschaften und Formatvorlagen (Style-Sheets) geändert/angepasst werden. Die allgemeinen Darstellungseigenschaften werden über die folgende Datei initialisiert:

- *CitizenWorkspace\Data_Manager\Initial_Data\blob\css\cp-config.properties*

Der Anwendungsressourcenname 'cp-config-properties' verweist auf diese Datei.

Die Hauptformatvorlage wird über die folgende Datei initialisiert:

- *CitizenWorkspace\Data_Manager\Initial_Data\blob\css\cp-css-template.css*

Der Anwendungsressourcenname 'cp-css-template' verweist auf diese Datei.

Das Banner wird in gleicher Weise über die folgende Datei initialisiert:

- *CitizenWorkspace\Data_Manager\Initial_Data\blob\css\banner-css-template.css*

Der Anwendungsressourcenname 'banner-css-template' verweist auf diese Datei.

Notieren Sie die Verwendung von Eigenschaften in den .css-Dateien wie beispielsweise 'banner.icon'. Wenn Universal Access die Formatvorlage lädt, werden diese Eigenschaften aus 'cp-config.properties' in der Schablone ersetzt, um die tatsächliche Formatvorlage zu erstellen, sodass viele Aspekte der Darstellung der Seitenwiedergabe einfach durch Ändern dieser Eigenschaftendatei geändert werden können, ohne die .css-Dateien ändern zu müssen. Wie bei den vorherigen Beispielen in diesem Abschnitt können die CSS-Vorlagen (css-templates) und zugehörigen Eigenschaften geändert werden, indem eine Kopie der DMX-Daten der Anwendungsressourcen in die angepasste Komponente kopiert werden.

Bitte lesen Sie im Kapitel zur Anpassung des Bürgerkontos die Informationen zum Anpassen der Darstellung und Funktionsweise von Bürgerkontoseiten.

Allgemeine Universal Access-Einstellungen

Die Datei

- *CitizenWorkspace\Data_Manager\Initial_Data\blob\prop\CPPagePlayer*.properties*

und die entsprechenden übersetzten Dateien (wie beispielsweise CPPagePlayer_es-.properties) steuern allgemeine Texte und Bilder, die mit der Universal Access-Anwendung in Zusammenhang stehen. Dies gilt beispielsweise für den Text der Schaltflächen 'Next' und 'Back' (auf deutsch 'Weiter' und 'Zurück'), für Texte in Seitenbannern etc. Diese Ressource wird unter dem Ressourcennamen 'CPPagePlayer' registriert und kann auf dieselbe Art und Weise geändert werden, wie in den vorstehenden Abschnitten zu Inhalts- und Hilfetexten beschrieben.

Anpassbare öffentliche Universal Access-APIs

Die Komponenten für den Citizen-Arbeitsbereich (CitizenWorkspace) und die Arbeitsbereichsservices (WorkspaceServices) enthalten Anwendungsprogrammierschnittstellen (APIs). Die Javadoc-Dokumentation für diese APIs befindet sich jeweils im folgenden Unterverzeichnis 'doc':

- *<Curam-Verzeichnis>\EJBServer\components\CitizenWorkspace\doc*
- *<Curam-Verzeichnis>\EJBServer\components\WorkspaceServices\doc*

Eine begrenzte Anzahl dieser APIs kann über Ereignis- und Strategiemuster angepasst werden. Die entsprechende Vorgehensweise wird im Handbuch zur Cúram-Entwicklungskonformität (Cúram Development Compliancy Guide) beschrieben.

Erweiterbare Codetabellen

Kunden wird empfohlen, die Liste der eingeschränkten Codetabellen im Handbuch zur Cúram-Entwicklungskonformität (Cúram Development Compliancy Guide) zu Rate zu ziehen.

Universal Access-Artefakte mit eingeschränkten Möglichkeiten für die Anpassung

Einführung

In diesem Kapitel werden die in Universal Access vorhandenen Artefakte mit eingeschränktem Zugriff beschrieben. Kunden, die Artefakte ändern wollen, die in diesem Kapitel aufgeführt sind, sollten über Alternativen nachdenken oder eine Erweiterung für Universal Access anfordern.

Modell

Änderungen am Universal Access-Modell im Ganzen oder in Teilen durch den Kunden werden nicht unterstützt. Durch Änderungen am Modell (beispielsweise das Ändern der Datentypen von Domänen) werden mit großer Wahrscheinlichkeit Störungen im Universal Access-System und/oder Upgradeprobleme verursacht. Dies gilt für die Modelldateien in den folgenden Paketen:

- WorkspaceServices (Arbeitsbereichsservices)
- CitizenWorkspace (Citizen-Arbeitsbereich)
- CitizenWorkspaceAdmin (Verwaltung des Citizen-Arbeitsbereichs)

Universal Access - XML-Dateien für Seitenwiedergabe

Änderungen an den XML-Dateien für Seitenwiedergabe durch den Kunden werden nicht unterstützt.

JSP- und JSPX-Seiten

Das Ändern der standardmäßig bereitgestellten JSP- oder JSPX-Dateien durch den Kunden wird nicht unterstützt.

JavaScript-Dateien

Das Ändern der in den folgenden Komponenten standardmäßig bereitgestellten JavaScript-Dateien durch den Kunden wird nicht unterstützt:

- WorkspaceServices (Arbeitsbereichsservices)
- CitizenWorkspace (Citizen-Arbeitsbereich)
- CitizenWorkspaceAdmin (Verwaltung des Citizen-Arbeitsbereichs)

Renderer - Konfiguration

Änderungen an den Renderer-Konfigurationsdateien durch den Kunden werden nicht unterstützt.

Dies gilt für die XML-Dateien an den folgenden Positionen:

- webclient\components\WorkspaceServices\Configuration
- webclient\components\CitizenWorkspace\Configuration
- webclient\components\CitizenWorkspaceAdmin\Configuration

Clientseitige Java-Artefakte

Universal Access stellt alle eigenen clientseitigen Java-Artefakte über die Datei 'CitizenWorkspace_source.jar' bereit. Diese JAR-Datei enthält sämtliche Klassen, die für Universal Access-Renderer und -Servlets erforderlich sind. Das Erweitern, Ändern und Ersetzen dieser bereitgestellten Klassen durch den Kunden wird nicht unterstützt.

Codetabellen

Kunden wird empfohlen, die Liste der eingeschränkten Codetabellen im Handbuch zur Cúram-Entwicklungskonformität (Cúram Development Compliancy Guide) zu Rate zu ziehen.

SOAP-Beispielanforderungen

Aktualisierung des Antrags für das Anliegenprogramm

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices
.curam" xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>userforpeersystem</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:updateIntakeProgramApplication>
      <rem:xmlMessage>
      <intakeProgramApplicationUpdate>
        <applicationReference>256</applicationReference>
        <applicationProgramReference>joannesprogram
</applicationProgramReference>
        <programApplicationStatus>IPAS1004</programApplicationStatus>
        <programApplicationDisposedDateTime>
          20120528 17:19:47
        </programApplicationDisposedDateTime>
        <programApplicationDenialReason>IPADR1001
</programApplicationDenialReason>
      </intakeProgramApplicationUpdate>
         </rem:xmlMessage>
      </rem:updateIntakeProgramApplication>
   </soapenv:Body>
</soapenv:Envelope>
```

Aktualisierung für Zurückziehungsantrag

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

  <column name="
      searchServiceFieldId
      " type="text" />
  <column name="
      searchServiceId
      " type="text" />
  <column name="
      name
      " type="text" />
  <column name="
      indexed
      " type="bool" />
  <column name="
      type
      " type="text" />
  <column name="
      stored
      " type="bool" />
  <column name="
      entityName
      " type="text" />
  <column name="
```

```
      analyzerName
      " type="text" />
  <column name="
      untokenized
      " type="bool" />

  <row>
    <attribute name="searchServiceFieldId">
      <value>
      field0
      </value>
    </attribute>
    <attribute name="searchServiceId">
      <value>
      PersonSearch
      </value>
    </attribute><attribute name="name">
      <value>
      primaryAlternateID
      </value>
    </attribute><attribute name="indexed"> <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:rem="http://remote.externalservices.workspaceservices.curam"
xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>userforpeersystem</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:updateWithdrawalRequest>
         <rem:xmlMessage>
    <withdrawalRequestUpdate>
      <curamReferenceID>-6897262829317914624</curamReferenceID>
      <withdrawalRequestStatus>WREQ1002</withdrawalRequestStatus>
      <resolvedDateTime>20120525 11:30:50</resolvedDateTime>
    </withdrawalRequestUpdate>
         </rem:xmlMessage>
      </rem:updateWithdrawalRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Konto erstellen

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:createAccount>
         <!--Optional:-->
         <rem:xmlMessage>
            <!--Optional:-->
     <cre:AccountCreate xmlns:cre="http://www.curamsoftware.com/
WorkspaceServices/ExternalAccountCreate">
        <firstName>John</firstName>
        <middleName>M</middleName>
        <surname>Doe</surname>
        <username>johnmdoe</username>
        <password>password1</password>
        <confirmPassword>password1</confirmPassword>
```

```
        <secretQuestionType>SQT1</secretQuestionType>
        <answer>mypassword1</answer>
        <termsAndConditionsAccepted>true</termsAndConditionsAccepted>
        <intakeApplicationReference>256</intakeApplicationReference>
        <clientIDOnRemoteSystem>112233445566</clientIDOnRemoteSystem>
        <sourceSystem>TestSystem</sourceSystem>
    </cre:AccountCreate>
         </rem:xmlMessage>
      </rem:createAccount>
   </soapenv:Body>
</soapenv:Envelope>
```

Kontoverknüfung

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:linkTargetSystemToAccount>
         <rem:xmlMessage>
          <lnk:AccountLink xmlns:lnk="http://www.curamsoftware.com/
WorkspaceServices/ExternalAccountLink">
           <sourceSystem>TestSystem</sourceSystem>
           <citizenWorkspaceAccountID>7081910414040104960
</citizenWorkspaceAccountID>
           <clientIDOnRemoteSystem>112233445566</clientIDOnRemoteSystem>
          <createdByUsername>testuser</createdByUsername>
         </lnk:AccountLink>
         </rem:xmlMessage>
      </rem:linkTargetSystemToAccount>
   </soapenv:Body>
</soapenv:Envelope>
```

Kontoverknüpfung aufheben

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:unlinkTargetSystemFromAccount>
         <!--Optional:-->
         <rem:xmlMessage>
         <unl:AccountUnlink xmlns:unl="http://www.curamsoftware.com/
WorkspaceServices/ExternalAccountUnlink">
           <sourceSystem>TestSystem</sourceSystem>
           <citizenWorkspaceAccountID>7081910414040104960
</citizenWorkspaceAccountID>
         </unl:AccountUnlink>
         </rem:xmlMessage>
      </rem:unlinkTargetSystemFromAccount>
   </soapenv:Body>
</soapenv:Envelope>
```

Bürgernachricht

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
    <soapenv:Header>
  <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:createMessage>
         <rem:xmlMessage>
<cm:CitizenMessage xmlns:cm="http://www.curamsoftware.com/
WorkspaceServices/ExternalCitizenMessage">
  <sourceSystem>TestSystem</sourceSystem>
  <cityIndustryType>CMI9001</cityIndustryType>
  <citizenWorkspaceAccountID>7081910414040104960
</citizenWorkspaceAccountID>
  <relatedID>6060</relatedID>
  <externalCitizenMessageType>PMT2004</externalCitizenMessageType>
  <messageTitle>Hello, World!</messageTitle>
  <messageBody>This is the body of the message.</messageBody>
  <effectiveDate>2000-01-01</effectiveDate>
  <expiryDate>2020-01-01</expiryDate>
  <priority>false</priority>

</cm:CitizenMessage>
         </rem:xmlMessage>
      </rem:createMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Zahlung (Einfach)

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:create>
         <rem:xmlMessage>
     <tns:Payment xmlns:tns="http://www.curamsoftware.com/
WorkspaceServices/ExternalPayment">
        <paymentID>1554</paymentID>
        <sourceSystem>TestSystem</sourceSystem>
        <cityIndustryType>CMI9001</cityIndustryType>
        <citizenWorkspaceAccountID>7081910414040104960
</citizenWorkspaceAccountID>
        <paymentAmount>50.00</paymentAmount>
        <currency>EUR</currency>
        <paymentMethod>CHQ</paymentMethod>
        <paymentStatus>PRO</paymentStatus>
        <effectiveDate>2012-01-01</effectiveDate>
        <coverPeriodFrom>2012-01-01</coverPeriodFrom>
        <coverPeriodTo>2012-01-01</coverPeriodTo>
        <dueDate>2012-01-01</dueDate>
        <payeeName>Dorothy</payeeName>
        <payeeAddress>12 Gloster St., WA 6008</payeeAddress>
        <paymentReferenceNo>F</paymentReferenceNo>
        <bankSortCode>933384</bankSortCode>
```

```
        <bankAccountNo>88776655</bankAccountNo>
        <PaymentBreakdown>
           <PaymentLineItem>
                <caseName>I</caseName>
                <caseReferenceNo>J</caseReferenceNo>
                <componentType>C10</componentType>
                <debitAmount>22.45</debitAmount>
                <creditAmount>50.76</creditAmount>
                <coverPeriodFrom>2012-01-01</coverPeriodFrom>
                <coverPeriodTo>2012-01-01</coverPeriodTo>
            </PaymentLineItem>
        </PaymentBreakdown>
    </tns:Payment>
  </rem:xmlMessage>
</rem:create>
</soapenv:Body>
</soapenv:Envelope>
```

Zahlung (Stapelverarbeitung)

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:create>
         <rem:xmlMessage>
     <tns:Payments xmlns:tns="http://www.curamsoftware.com/
WorkspaceServices/ExternalPayment">
        <Payment>
            <paymentID>2346</paymentID>
            <sourceSystem>TestSystem</sourceSystem>
            <cityIndustryType>CMI9001</cityIndustryType>
            <citizenWorkspaceAccountID>8306889512684879872
</citizenWorkspaceAccountID>
            <paymentAmount>48.00</paymentAmount>
            <currency>EUR</currency>
            <paymentMethod>CHQ</paymentMethod>
            <paymentStatus>PRO</paymentStatus>
            <effectiveDate>2012-01-01</effectiveDate>
            <coverPeriodFrom>2012-01-01</coverPeriodFrom>
            <coverPeriodTo>2012-01-01</coverPeriodTo>
            <dueDate>2012-01-01</dueDate>
            <payeeName>D</payeeName>
            <payeeAddress>E</payeeAddress>
            <paymentReferenceNo>F</paymentReferenceNo>
            <bankSortCode>G</bankSortCode>
            <bankAccountNo>H</bankAccountNo>
            <PaymentBreakdown>
                <PaymentLineItem>
                    <caseName>I</caseName>
                    <caseReferenceNo>J</caseReferenceNo>
                    <componentType>C24000</componentType>
                    <debitAmount>22.45</debitAmount>
                    <creditAmount>49.76</creditAmount>
                    <coverPeriodFrom>2012-01-01</coverPeriodFrom>
                    <coverPeriodTo>2012-01-01</coverPeriodTo>
                </PaymentLineItem>
                <PaymentLineItem>
                    <caseName>I</caseName>
                    <caseReferenceNo>J</caseReferenceNo>
                    <componentType>C24000</componentType>
```

```
                    <debitAmount>22.45</debitAmount>
                    <creditAmount>49.76</creditAmount>
                    <coverPeriodFrom>2012-01-01</coverPeriodFrom>
                    <coverPeriodTo>2012-01-01</coverPeriodTo>
                </PaymentLineItem>
            </PaymentBreakdown>
        </Payment>
    </tns:Payments>
         </rem:xmlMessage>
      </rem:create>
   </soapenv:Body>
</soapenv:Envelope>
```

Kontaktperson

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
   <soapenv:Header>
  <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:updateExternalContact>
         <rem:xmlMessage>
     <con:ContactInfo xmlns:con="http://www.curamsoftware.com/
WorkspaceServices/ExternalContact">
        <sourceSystem>TestSystem</sourceSystem>
        <contactReference>CON_100</contactReference>
         <fullName>Harry Neilan</fullName>
        <phoneNumber>1-800-CALL-ME</phoneNumber>
        <mobilePhoneNumber>1-800-CALL-MOB</mobilePhoneNumber>
        <faxNumber>1-800-CALL-FAX</faxNumber>
        <email>harry@x.org</email>
    </con:ContactInfo>
         </rem:xmlMessage>
      </rem:updateExternalContact>
   </soapenv:Body>
</soapenv:Envelope>
```

Fälle

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:rem="http://remote.externalservices.workspaceservices.
curam" xmlns:xsd="http://dom.w3c.org/xsd">
<soapenv:Header>
     <curam:Credentials xmlns:curam="http://www.curamsoftware.com">
     <Username>admin</Username>
     <Password>password</Password>
   </curam:Credentials>
   </soapenv:Header>
   <soapenv:Body>
      <rem:updateExternalCase>
         <rem:xmlMessage>
    <cas:CaseInfo xmlns:cas="http://www.curamsoftware.com/
WorkspaceServices/ExternalCase">
        <sourceSystem>TestSystem</sourceSystem>
        <contactReference>CON_100</contactReference>
        <caseReference>CAS_109</caseReference>
        <caseName>My Benefit Case - 103</caseName>
        <citizenWorkspaceAccountID>8306889512684879872
</citizenWorkspaceAccountID>
    </cas:CaseInfo>
```

```
            </rem:xmlMessage>
         </rem:updateExternalCase>
      </soapenv:Body>
   </soapenv:Envelope>
```

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden. Für die in diesem Handbuch beschriebenen Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Défense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht. Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
Dept F6, Bldg 1
294 Route 100
Sommers NY 10589-3216
U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen.

IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Alle von IBM angegebenen Preise sind empfohlene Richtpreise und können jederzeit ohne weitere Mitteilung geändert werden. Händlerpreise können u. U. von den hier genannten Preisen abweichen.

Diese Veröffentlichung dient nur zu Planungszwecken. Die in dieser Veröffentlichung enthaltenen Informationen können geändert werden, bevor die beschriebenen Produkte verfügbar sind.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Beispielprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM übernimmt keine Haftung für Schäden, die durch die Verwendung der Beispielprogramme entstehen.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name Ihrer Firma) (Jahr). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet.

© Copyright IBM Corp. _enter the year or years_. Alle Rechte vorbehalten.

Hinweise zur Datenschutzrichtlinie

IBM Softwareprodukte, einschließlich Software as a Service-Lösungen ("Softwareangebote"), können Cookies oder andere Technologien verwenden, um Informationen zur Produktnutzung zu erfassen, die Endbenutzererfahrung zu verbessern und Interaktionen mit dem Endbenutzer anzupassen oder zu anderen Zwecken. In vielen Fällen werden von den Softwareangeboten keine personenbezogenen Daten erfasst. Einige der IBM Softwareangebote können Sie jedoch bei der Erfassung personenbezogener Daten unterstützen. Wenn dieses Softwareangebot Cookies zur Erfassung personenbezogener Daten verwendet, sind nachfolgend nähere Informationen über die Verwendung von Cookies durch dieses Angebot zu finden.

Je nachdem, welche Konfigurationen implementiert wurden, ist es möglich, dass dieses Softwareangebot Sitzungscookies und persistente Cookies zum Erfassen der Namen, Benutzernamen, Kennwörter, Profilnamen oder anderer personenbezogener Daten einzelner Benutzer für die Sitzungsverwaltung, Authentifizierung, Single-Sign-on-Konfiguration oder für einen besseren Bedienungskomfort und/oder andere Zwecke der Nutzungsverfolgung bzw. funktionale Einsatzmöglichkeiten. Diese Cookies oder ähnliche Technologien können nicht inaktiviert werden.

Wenn die für dieses Softwareangebot genutzten Konfigurationen Sie als Kunde in die Lage versetzen, personenbezogene Daten von Endbenutzern über Cookies und andere Technologien zu erfassen, müssen Sie sich zu allen gesetzlichen Bestimmungen in Bezug auf eine solche Datenerfassung, einschließlich aller Mitteilungspflichten und Zustimmungsanforderungen, rechtlich beraten lassen.

Weitere Informationen zur Nutzung verschiedener Technologien, einschließlich Cookies, für diese Zwecke finden Sie in der "IBM Online-Datenschutzerklärung, Schwerpunkte" unter http://www.ibm.com/privacy und in der "IBM Online-Datenschutzerklärung" unter http://www.ibm.com/privacy/details im Abschnitt "Cookies, Web-Beacons und sonstige Technologien" und unter "IBM Software Products and Software-as-a-Service Privacy Privacy Statement" unter http://www.ibm.com/software/info/product-privacy.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation.Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information" unter http://www.ibm.com/legal/us/en/copytrade.shtml.

Adobe, das Adobe-Logo und das Portable Document Format (PDF) sind Marken oder eingetragene Marken der Adobe Systems Incorporated in den USA und/oder anderen Ländern.

Apache ist eine eingetragene Marke der Apache Software Foundation.

WebLogic Server, Java und alle auf Java basierenden Marken und Logos sind eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Andere Namen können Marken der jeweiligen Rechtsinhaber sein. Weitere Firmen-, Produkt- und Servicenamen können Marken oder Servicemarken anderer Unternehmen sein.