# Plugin Status Chart Toolkit (SCT) - Technische Dokumentation

Das Plugin Status Chart Toolkit (SCT) ist ein APEX Dynamic-Action Plugin zur Erstellung komplexer APEX-Formulare ohne den ansonsten notwendigen Overhead vieler Dynamic Actions zur Verwaltung des Status der Formularelemente auf einer APEX Seite.

Es basiert auf der Extraktion der Entscheidungslogik, welches Formularelement welchen Sichtbarkeitsstatus und Wert haben soll, in eine Regeltabelle innerhalb der Datenbank. Auf diesem Weg wird diese Logik zentralisiert und dadurch übersichtlicher, zudem lassen sich Seiteneffekte der Regeln einfacher detektieren, weil das Plugin dem Entwickler Informationen über die Regel gibt, die für einen konkreten Status der Anwendung verwendet wurde.

Das Plugin hat folgenden Funktionsumfang:

* Verwaltung des Sichtbarkeitsstatus von Formularelementen (aktiv, deaktiv, sichtbar, versteckt)
* Validierung der Eingaben, Ausgabe entsprechender Fehlermeldungen
* Verwaltung des Session State, Ermittlung von Elementwerten basierend auf der Auswahl des Benutzers der Anwendung
* Rekursive Regelauslösung
* Einfache Administration: Das Plugin erfordert keinerlei Administration auf der Anwendungsseite, alle notwendigen Eventhandler, die Auswahl der zu übertragenden Elementwerte und alle weitere Konfiguration wird automatisch aus den eingegebenen Regeln abgeleitet

Alle Funktionen des Plugins werden dynamisch über AJAX-Aufrufe angeboten, ein erneutes Laden der Anwendungsseite erfolgt nicht.

## Konzept des Plugins

Das Plugin basiert auf der Tatsache, dass komplexe Fallunterscheidungen immer auch in SQL als WHERE-Klausel ausgedrückt werden können. Anstatt eine Fallunterscheidung diskret, z.B. in PL/SQL oder JavaScript zu programmieren, werden die Regeln als Fragmente einer WHERE-Klausel ausgedrückt und in einer Datenbanktabelle gespeichert. Eine Regel besteht, neben der Bedingung, aus einer Sammlung von Aktionen, die ausgeführt werden sollen, wenn diese Regel angewendet werden soll. Diese Aktionen können entweder innerhalb der Datenbank als PL/SQL-Aktionen, oder im Browser, als JavaScript-Aktionen, ausgeführt werden. Regeln werden in der Datenbank in Regelgruppen organisiert, deren anwendungsweit eindeutiger Regelgruppenname durch das Plugin referenziert wird. Dadurch ist es möglich, Regeln in Gruppen zu organisieren und eine oder mehrere Regelgruppen auf einer Anwendungsseite zu definieren. Für jede Regelgruppe muss auf der Anwendungsseite eine Instanz des Plugins angelegt werden.

Das Plugin liest beim Aufruf den aktuellen Zustand aller »relevanter« Seitenelemente im Browser und stellt diese Werte einer SQL-Abfrage zur Verfügung, die anschließend durch die, in der WHERE-Klausel enthaltenen, Regeln geprüft werden. Die erste Regel gemäß eines Sortierkriteriums, das bei der Regeldefinition festgelegt wird, die zu TRUE evaluiert, legt das weitere Verhalten des Plugins fest, indem die Aktionen dieser Regel ausgeführt bzw. im Browser veranlasst werden.

Als »relevant« werden im Plugin die Seitenelemente betrachtet, die in einer Einzelregel der Regelgruppe angesprochen werden. Ist also eine Einzelregel definiert als P1\_VALID\_FROM < P1\_VALID\_TIL, extrahiert das Plugin daraus die beiden Elemente P1\_VALID\_FROM und P1\_VALID\_TIL als relevant. Extrahierte Elemente müssen auf der Anwendungsseite verwendet worden sein, oder als Anwendungselemente zur Verfügung stehen. Wird ein Element verwendet, dass nicht vorhanden ist, validiert die Regel nicht.

Dass die relevanten Elemente extrahiert werden, hat zur Folge, dass ...

* diese beiden Elemente durch das Plugin mit einem Eventhandler für den CHANGE-Event (oder CLICK-Event, wenn es sich um eine Schaltfläche handelt) versehen werden. Dadurch wird die Verarbeitung der geänderten Eingaben dieser Elemente oder der Klick auf die Schaltfläche durch das Plugin ausgelöst
* beim Aufruf des Plugins die aktuellen Elementwerte dieser Elemente in den Session State der APEX-Anwendung kopiert werden. Dadurch stehen die aktuellen Elementwerte der SQL-Regelabfrage zur Verfügung

Sollte das Plugin Daten der Anwendungselemente im Session State ändern, werden diese in der Antwort des Plugins an die Anwendung übergeben und durch das Plugin in die entsprechenden Formularelemente eingefügt. Es können nicht nur »relevante«, sondern alle Formularelemente der Seite durch das Plugin gesetzt werden.

## Technische Umsetzung: Module

Das Plugin besteht aus mehreren Modulen:

* Kernfunktionalität  
  Implementiert in den Packages SCT\_ADMIN und SCT\_CONS
* Plugin  
  Das eigentliche APEX-Plugin, implementiert im Package PLUGIN\_SCT sowie den JavaScript-Dateien SCT.js und SCTApex.js
* APEX-Anwendung  
  APEX-Anwendung zur Verwaltung der Regelgruppen, Einzelregeln, Aktionen und Aktionstypen, implementiert im Package UI\_SCT\_PKG sowie der APEX-Anwendung selbst

### Kernfunktionalität: Package SCT\_ADMIN

Das Package SCT\_ADMIN enthält die zentrale Logik des Projekts, die hier definierten Methoden werden einerseits direkt, andererseits durch die Packages PLUGIN\_SCT sowie UI\_SCT\_PKG aufgerufen. Das Package umfasst mehrere Funktionsbereiche:

* Administration  
  Erstellung und Verwaltung von Regelgruppen, Einzelregeln, Aktionen und Aktionstypen, Methoden zum Im- und Export von Regelgruppen zur Integration des Plugins in einen APEX-Deploymentzyklus
* Interne Verwaltung  
  Automatisierte Definition der Regelabfragen, Prüfung von Regeln
* Anwendung des Plugins  
  Methode zur Analyse einer Regelgruppe und zur Erstellung der PL/SQL- und JavaScript-Skripte als Antwort des Plugins auf eine gegebene Anwendungssituation, Bereitstellung von Hilfsfunktionen für die Package PLUGIN\_SCT und UI\_SCT\_PKG

#### Getter-Methoden GET\_FIRING\_ITEM, GET\_JS\_FUNCTION

Die Methoden liefern den Namen des auslösenden Elements sowie den Namen der JavaScript-Funktion, mit der das Plugin aufgerufen wird. Sie werden von der Regelview und vom Package PLUGIN\_SCT aufgerufen.

#### Methode COPY\_RULE\_GROUP

Die Methode kopiert eine existierende Regelgruppe. Ziel kann die gleiche oder eine andere Anwendung sein. Die Methode wird verwendet, um bestehende Regeln bei Neuanlage einer Anwendungsseite unter einer anderen ID in derselben Anwendung migrieren oder duplizieren zu können, sowie um eine Regelgruppe zwischen Anwendungen zu kopieren.

#### MERGE-Methoden

Die Methoden dienen dem Erstellen oder Ändern bestehender Regelgruppen, Regeln, Aktionen oder Aktionstypen. Sie werden nicht von der APEX-Anwendung verwendet, sondern dienen dem Import von Regelgruppen in eine Anwendung.

#### Methoden EXPORT\_RULE\_GROUP und EXPORT\_RULE\_GROUPS

Die Methode EXPORT\_RULE\_GROUP ist überladen, um den Export einer Regelgruppe nach ID oder Regelgruppenname und Anwendungs-ID zu ermöglichen. Methode EXPORT\_RULE\_GROUPS ermöglicht den Export aller Regelgruppen einer Anwendung. Die Methoden liefern die Exportskripte als CLOB zurück, die Anwendung verfügt über eine Wartungsseite, mit deren Hilfe die Exporte ausgeführt und als Datei geladen werden können.

Beim Export einer Regelgruppe kann festgelegt werden, ob der Export für eine andere Anwendung und/oder Anwendungsseite erfolgen soll. Dadurch kann die Export-Funktion genutzt werden, um Regeln zwischen Anwendungen zu kopieren, oder aber auch, um Regeln beim Export einer APEX-Anwendung in den Skript zur Installation zu integrieren. Existiert die Regelgruppe für die angeforderte Anwendungs-ID und Seiten-ID, überschreibt der Export die existierende Regelgruppe.

#### Methode DELETE\_RULE\_GROUP

Die Methode entfernt die angegebene Regelgruppe aus den Stammdatentabellen.

#### Methode PROPAGATE\_RULE\_CHANGE

Die Methode PROPAGATE\_RULE\_CHANGE wird nach jeder Änderung einer Regel oder Regelgruppe aufgerufen. Durch den Aufruf der Methode wird veranlasst, dass ...

* Die Regel validiert wird
* Die Liste der relevanten Elemente der Regelgruppe neu kalkuliert wird
* Eine neue Version der Regelview erstellt wird

Die Methode wird durch die APEX-Anwendung zur Pflege des Plugins nach jeder Änderung aufgerufen. Die APEX-Anwendung basiert auf Assistenten-erstellen Seiten zur Pflege der Stammdaten, daher werden auch die Standardprozesse zum Anlegen und Speichern der Daten verwendet. Um einen Trigger zu vermeiden, der die Validierung der Regeländerung anstößt, wird auf den relevanten Seiten der APEX-Anwendung diese Methode explizit nach den Standardprozessen der Seite aufgerufen.

#### Methode CREATE\_ACTION

Diese Methode berechnet, welche Regel für eine gegebene Situation im Session State relevant ist, sowie die daraus resultierenden Aktionen für die Datenbank und die Anwendung. Aus der ermittelten Regel berechnet die Methode einen anonymen PL/SQL-Block mit den PL/SQL-Anweisungen sowie eine Liste mit JavaScript-Funktionen für die Anwendung. Die Ausführung des PL/SQL-Codes und die Aufbereitung der Antwort für die APEX-Anwendung erfolgen im Package PLUGIN\_SCT.

#### Methode RESEQUENCE\_RULES

Die Methode stellt alle Spalten SORT\_SEQ der Stammdatentabellen SCT\_RULE und SCT\_RULE\_ACTION auf eine aufeinanderfolgende Reihe in 10er-Spreizung zurück, um die Übersichtlichkeit zu erhöhen und weitere Sortierungen zu ermöglichen. Sie wird durch die APEX-Anwendung aufgerufen.

### Kernfunktionalität: Package SCT\_CONS

Da im Package SCT\_ADMIN viele SQL-Template erforderlich sind, um

* Exportskripte zu erzeugen
* Regelviews zu erstellen
* Regeln zu validieren

und diese Templates den Lesefluss des Packages SCT\_ADMIN stören, wurden diese in ein Konstantenpackage ausgelagert. SCT\_CONS enthält daher lediglich diese Templates, eine Implementierung benötigt dieses Package nicht.

### Plugin: Parameter

Das Plugin definiert zwei Parameter, die beim Erstellen einer Plugin-Instanz auf einer Anwendungsseite gefüllt werden:

* Name der Regelgruppe  
  Dieser Parameter referenziert den Namen der Regelgruppe, der in der Datenbank hinterlegt ist
* Fehlerbezogene Elementliste  
  Dieser Parameter ist optional und umfasst eine kommaseparierte Liste von Elementnamen, die durch das Plugin auf der Anwendungsseite deaktiviert werden, solange die Seite Fehlermeldungen zeigt, und aktiviert werden, sobald keine Fehlermeldungen auf der Seite angezeigt werden.

### Plugin: Package PLUGIN\_SCT

Das Package PLUGIN\_SCT implementiert die, durch APEX vorgegebene, Schnittstelle für ein Dynamic Action Plugin, namentlich die Methoden RENDER und AJAX. Zudem umfasst das Plugin weitere Methoden zum Setzen des Session State (SET\_SESSION\_STATE) sowie zum Registrieren von Fehlern, die während der Verarbeitung der Plugin-Anfrage innerhalb der Datenbank aufgetreten sind (REGISTER\_ERROR). Zudem besteht die Möglichkeit, durch das Package Benachrichtigungen auf die APEX-Anwendungsseite zu bringen. Auch hierfür lassen sich Meldungen durch die Methode REGISTER\_NOTIFICATION im Plugin registrieren.

#### Methode RENDER

Die RENDER-Methode des Packages initialisiert das Plugin, indem es für eine Regelgruppe eine Liste der relevanten Formularelemente erstellt, an die das Package im Anschluss Eventhandler bindet. Gebunden werden können CHANGE-Events an Formularelemente und CLICK-Events an Formularschaltflächen. Wie bereits beschrieben, erfolgt die Auswahl der Elemente, an die das Plugin bindet, automatisch über eine Analyse der definierten Regeln.

Der Ablauf des Renderprozesses ist wie folgt:

* Das Plugin wird, basierend auf den Parameterwerten des Plugins auf der Anwendungsseite, erstellt und auf der Seite eingerichtet
* Die Datenbank ermittelt alle relevanten Elemente der Regelgruppe und gibt diese als Parameter an das Plugin weiter
* Das Plugin bindet entsprechende Event-Handler an die Events, die aus der Datenbank für den entsprechenden Elementtyp hinterlegt sind (Formularelemente binden an den CHANGE-Event, Schaltflächen an den CLICK-Event, siehe Tabelle SCT\_PAGE\_ITEM\_TYPE)
* Zum Abschluss der Initialisierung ruft das Plugin erstmalig die AJAX-Funktion des Plugins auf und übergibt
  + Den aktuellen Elementwert aller relevanter Elemente
  + Die Angabe DOCUMENT als auslösendes Element

Anschließend wird mit diesem Aufruf verfahren wie im nächsten Abschnitt beschrieben.

#### Methode AJAX

Die AJAX-Methode implementiert das Verhalten des Plugins, wenn ein gebundener Event auf der Anwendungsseite ausgelöst wird.

Die JavaScript-Funktionalität des Plugins stellt die aktuellen Elementwerte der relevanten Elemente zusammen und persistiert diese im Session State und übermittelt den Namen des auslösenden Elements an die AJAX-Methode. Nach der Initialisierung des Plugins beim Seitenladen wird erstmalig diese Funktion mit dem auslösenden Elemente DOCUMENT aufgerufen, um sicherzustellen, dass die Anwendungsseite korrekt initialisiert wird.

In der AJAX-Methode werden folgende Schritte veranlasst:

* Ein interner Errorstack wird initialisiert.  
  Auf diesem Errorstack werden die eingehenden Fehlermeldungen durch die Methode REGISTER\_ERROR abgelegt
* Ein Rekursionsstack wird initialisiert.  
  Der Rekursionsstack ist eine PL/SQL-Tabelle, die als Schlüsselwerte die ID des auslösenden Elements verwendet und als Ladung die Rekursionstiefe, die bei 1 beginnt
* Die Regelgruppe wird analysiert.  
  Das Plugin-Package selbst enthält keine Entscheidungslogik und kann Regeln nicht evaluieren. Diese Aufgabe delegiert das Package an SCT\_ADMIN und enthält von dort Angaben darüber, welche Aktionen in PL/SQL innerhalb der Datenbank und in JavaScript im Browser auszuführen sind. Ist die Regelgruppe ausgewertet, wird ein PL/SQL-Script und ein JavaScript-Script an das Package geliefert.
* Das Plugin führt die Datenbankaktionen aus und analysiert, ob durch diese Aktionen ein rekursiver Aufruf von Regeln erforderlich ist.  
  Ein rekursiver Aufruf wird erforderlich, wenn während der Ausführung des PL/SQL-Blocks die Methode SET\_SESSION\_STATE des Plugins aufgerufen wurde und das geänderte Element noch nicht im Rekursionsstack enthalten ist. Wird dies erkannt, wird das Element in den Rekursionsstack eingefügt und der Rekursionszähler inkrementiert.
* Befinden sich Einträge im Rekursionsstack, die evaluiert werden müssen, beginnt die Analyse der Regelgruppe für jeden Eintrag erneut. Dabei wird als auslösendes Element der Schlüssel des Rekursionsstacks verwendet. Es werden zunächst alle Einträge des Rekursionsstacks abgearbeitet und die PL/SQL- und JavaScript-Scripte entgegengenommen, bevor eine erneute Rekursionsrunde angestoßen wird.
* Sind alle rekursiven Aufrufe abgeschlossen, werden die geänderten Elemente und die gesammelten JavaScript-Anweisungen in eine JSON-formatierte Antwort integriert und an die APEX-Anwendungsseite gesendet. Fehlermeldungen werden ebenso integriert wie eine Liste der Elemente, die durch die konkrete Regel tangiert wurde. Diese Liste wird verwendet, um auf der Seite vorhandene Fehlermeldungen zu entfernen.
* Das Plugin führt die Anweisungen der Seite aus und schließt den Bearbeitungszyklus damit ab.

#### Methode SET\_SESSION\_STATE

Für die korrekte Funktion des Plugins ist es wichtig, Elementwerte im Session State ausschließlich über die Methode SET\_SESSION\_STATE zu setzen und nicht direkt über APEX\_UTL.SET\_SESSION\_STATE, um sicher zu stellen, dass das Plugin Kenntnis von der Werteänderung enthält und diese an die Anwendung weiterreichen kann.

Alle Elemente, die durch die Bearbeitung der Regelanfrage geändert werden, werden durch das Plugin registriert und in der Antwort mit ihren aktuellen Elementwerten aus dem Session State an die Anwendungsseite gereicht. Das Plugin setzt die Elementwerte, so dass Seite und Session State bezüglich der geänderten Elemente nachfolgend synchron sind.

#### Methode REGISTER\_ERROR

Die Methode REGISTER\_ERROR sammelt alle auflaufenden Fehler während der Verarbeitung und gibt diese an die JavaScript-Methoden des Plugins weiter, so dass diese dort bei den entsprechenden Elementen als Fehler angezeigt werden können. Ein Fehler wird über den Namen (ID) des Anwendungselements und den Fehlertext definiert. Durch die Angabe der ID des Anwendungselementes, dass den Fehler ausgelöst hat, ist das Plugin in der Lage, den Fehler bei dem betroffenen Element anzuzeigen und diesen selektiv von der Anwendungsseite zu löschen.

#### Methode REGISTER\_NOTIFICATION

Die Methode REGISTER\_NOTIFICATION sammelt, analog zur Methode REGISTER\_ERROR, Benachrichtigungen an den Anwendungsbenutzer, die während der Verarbeitung der Regel aufgelaufen sind. Nach Abschluss der Verarbeitung werden alle Regeln zusammengefasst und in einem Notification-Block auf der APEX-Seite dargestellt. Bei den Meldungen kann es sich um Bestätigungsmeldungen, Statusmeldungen oder ähnliche handeln.

### APEX-Anwendung: Anwendung

Die APEX-Anwendung des Plugins dient der vereinfachten Pflege von Regelgruppen, Einzelregeln, Aktionen und Aktionstypen. Zusätzlich zu den Grundfunktionen, also der Anlage, Pflege und dem Löschen der beschriebenen Elemente, verfügt die Anwendung über eine Validierungsfunktion, die Regeln gegen das Data Dictionary der APEX-Anwendung, für die die Regel erstellt wird, validiert.

Die Anwendung selbst ist mit einfachen APEX-Standardmitteln erstellt, die Stammdatentabellen werden direkt über Assistenten-erstellte Seiten gepflegt, nicht über eine Methodenschnittstelle. Die Methoden zur Anlage von Regelgruppen etc. werden ausschließlich von der Im- und Exportfunktionalität verwendet, nicht jedoch von der APEX-Anwendung. Eine Ausnahme hiervon sind die Funktionen zum Kopieren von Regelgruppen zwischen Anwendungen oder Anwendungsseiten sowie das Resequenzieren einer Regelgruppe. Zudem werden durch die Anwendung nach dem Speichern von Änderungen die Änderungen propagiert, d.h. es werden die Validierung sowie die Neuerstellung der Regelviews veranlasst.

Die Hauptseite der Anwendung verwendet selbst wiederum das Plugin, um Schaltflächen einzublenden, Berichte zu aktualisieren und weiteres mehr. Die hierfür benötigten Regeln werden in der Übersicht der Regelgruppen ausgeblendet, um ein versehentliches Löschen dieser Regeln zu vermeiden.

### APEX-Anwendung: Package UI\_SCT\_PKG

Das Package UI\_SCT\_PKG implementiert die Anwendungslogik für die APEX-Anwendung zur Verwaltung der Regelgruppen, Einzelregeln, Aktionen und Aktionstypen. Die eigentliche Logik delegiert das Package an SCT\_ADMIN. Daher stellen die Methoden des Packages lediglich Wrapper um die Methoden des Packages SCT\_ADMIN dar, zum Teil mit unterschiedlicher Parameterausstattung.

## Technische Umsetzung: Arbeitsweise

### Datenmodell

#### Regelgruppe, Tabelle SCT\_RULE\_GROUP

Soll auf einer APEX-Anwendungsseite die Kontrolle des Status der Formularelemente, Regionen und Schaltflächen durch das Plugin SCT übernommen werden, muss zunächst eine Regelgruppe für diese Seite angelegt werden. Eine Regelgruppe umfasst

* eine technische ID
* einen anwendungsweit eindeutigen Namen inkl. optionaler Beschreibung
* eine Referenz auf die APEX-Anwendung und -Seite, jeweils über die ID

Aufgabe der Regelgruppe ist ausschließlich die Gruppierung von Regeln. Sie stellt einen Zuständigkeitsbereich dar, innerhalb dessen Regeln und deren Abhängigkeiten analysiert werden. Regelgruppen werden in der Tabelle SCT\_RULE\_GROUP gespeichert.

#### Einzelregel, Tabelle SCT\_RULE

Der Regelgruppe werden im folgenden Einzelregeln zugeordnet, die wiederum aus einer technischen ID, einem beschreibenden Namen, einer Bedingung und einem Sortierkriterium bestehen. Die Einzelregeln referenzieren die Regelgruppen, zu denen sie gehören, über deren technische ID und dienen gleichzeitig als Referenz für beliebig viele Aktionen, die ausgeführt werden sollen, wenn die Bedingung der Einzelregel zu TRUE evaluiert und das Plugin diese Einzelregel zur Ausführung auswählt.

Zusätzlich pflegt das Plugin für jede Einzelregel eine Liste mit den Namen der Formularelemente oder -schaltflächen, die in der Bedingung referenziert werden. Die Liste dieser referenzierten Elemente stellt in ihrer Gesamtheit die Liste der »relevanten« Elemente dieser Regelgruppe dar. Einzelregeln werden in der Tabelle SCT\_RULE gespeichert.

#### Aktionen, Tabelle SCT\_RULE\_ACTION

Jeder Einzelregel werden Aktionen zugeordnet, die ausgeführt werden sollen, wenn die Regel zu TRUE evaluiert und das Plugin diese Einzelregel zur Ausführung auswählt. Die Aktionen referenzieren einen Aktionstyp, der in Tabelle SCT\_ACTION\_TYPE hinterlegt ist, sowie die Einzelregel und die Regelgruppe, zu der sie gehören. Zudem referenzieren sie ein Seitenelement, auf das sich die Aktion bezieht. Soll also z.B. der Wert des Formularelements P1\_VALID\_TIL auf den Wert NULL gesetzt werden, referenziert diese Aktion das Element P1\_VALID\_TIL und ruft den Aktionstyp SET\_NULL auf, der wiederum definiert, wie ein Elementwert auf den Wert NULL zu setzen ist. Auch die Aktionen verfügen über ein Sortierkriterium, das steuert, in welcher Reihenfolge Aktionen ausgeführt werden sollen.

Aktionen enthalten zwei optionale Attributwerte in den Spalten ATTRIBUTE und ATTRIBUTE\_2. Diese Attribute können durch die Aktionstypen referenziert werden (siehe dort) und dienen der Übermittlung von Parametern an den PL/SQL oder JavaScript-Code.

Aktionen werden in der Tabelle SCT\_RULE\_ACTION gespeichert.

#### Aktionstypen, Tabelle SCT\_RULE\_ACTION\_TYPE

Eine Aktion hat einen Aktionstyp. Dieser definiert, wie eine Aktion implementiert werden soll. Ein Aktionstyp umfasst einen technischen, alphanumerischen Schlüssel, einen Klarnamen zur Darstellung auf der Oberfläche, eine Implementierungsanweisung für die Sprache PL/SQL sowie eine Implementierungsanweisung für die Sprache JavaScript. Die Implementierungsanweisungen sind optional und können gar nicht, nur in PL/SQL, nur in JavaScript oder in beiden Sprachen gleichzeitig gesetzt sein. Anweisungen sind stets als Ausriss aus einem anonymen PL/SQL-Block oder einem JavaScript-Block zu schreiben, benötigen also insbesondere ein Semikolon an ihrem Ende.

Das Plugin sammelt alle PL/SQL-Anweisungen und führt diese innerhalb der Datenbank aus, anschließend werden die geänderten Elementwerte aus dem Session State geladen und mit den JavaScript-Anweisung an die APEX-Anwendung gesendet, wo sie ausgeführt werden. Daraus ergibt sich eine feste Ausführungsreihenfolge:

* Alle PL/SQL-Anweisungen in der Reihenfolge, die durch Spalte SORT\_SEQ der Tabelle SCT\_RULE\_ACTION definiert ist
* Alle JavaScript-Anweisungen in der Reihenfolge, die durch Spalte SORT\_SEQ der Tabelle SCT\_RULE\_ACTION definiert ist

Aktionstypen dürfen drei Ersetzungszeichenfolgen referenzieren, die bei der späteren Verwendung mit Bewegungsdaten gefüllt werden:

* #ITEM#  
  Dieser Ersetzungsanker wird zur Laufzeit mit dem Namen des referenzierten Objekts aus Spalte SRA\_SPI\_ID der Tabelle SCT\_RULE\_ACTION gefüllt
* #ATTRIBUTE# bzw. #ATTRIBUTE\_2#  
  Diese Ersetzungsanker werden zur Laufzeit mit den Spalten ATTRIBUTE bzw. ATTRIBUTE\_2 der Tabelle SCT\_RULE\_ACTION gefüllt. Sie dienen der Übergabe von Parametern an PL/SQL- oder JavaScript-Funktionen, z.B. für Meldungstexte etc.

Aktionstypen werden in der Tabelle SCT\_ACTION\_TYPE gespeichert.

#### Seitenelemente, Tabelle SCT\_PAGE\_ITEM

Regelbedingungen und Aktionen referenzieren Seitenelemente. Die referenzierten Seitenelemente werden aus dem APEX Data Dictionary für die APEX-Anwendung, die in der Regelgruppe referenziert wird, ausgelesen und analysiert. Die aufbereiteten Beschreibungen dieser Seitenelemente werden anschließend in Tabelle SCT\_PAGE\_ITEM gespeichert. Jedes Seitenelement wird beschrieben

* durch eine Referenz auf die Regelgruppe, von der das Seitenelement referenziert wird,
* durch die ID des Elements (Bei Formularelemente ist dies der Name des Elements, bei Schaltflächen und Regionen wird ausschließlich die statische ID, die auf der APEX-Anwendung vergeben wurde, referenziert),
* eine Referenz auf den Typ des Elements (Seitenelement, Anwendungselement, Schaltfläche oder Region), der in Tabelle SCT\_PAGE\_ITEM\_TYPE hinterlegt ist,
* sowie Angaben zur Konvertierung des Elements in die Datentypen TEXT, DATE oder NUMBER.

Die Tabelle umfasst lediglich Elemente, die in einer Regelbedingung referenziert wurden, denn diese Elemente werden für die Regelauswertung benötigt. Aktionen können alle Elemente der APEX-Anwendung ändern, doch führen diese Änderungen nur dann zu einer Reaktion des Plugins, wenn die geänderten Elemente in einer Regelbedingung verwendet wurden.

Die Konvertierung eines Elements in die Datentypen DATE oder NUMBER wird nur erkannt, wenn das Element auf der Anwendungsseite über eine Formatmaske verfügt. Daher müssen folgende Regeln beachtet werden, wenn das Plugin problemlos mit einer Anwendungsseite kooperieren können soll:

* Elemente, die in NUMBER oder DATE konvertiert werden sollen, müssen über eine Formatmaske verfügen
* Verfügen Seitenelemente in APEX nicht über die Möglichkeit, Formatmasken zu definieren, müssen diese Elemente in den Regelbedingungen explizit in die Zieldatentypen überführt werden (z.B. SELECT-Listen)
* Schaltflächen oder Regionen, die durch Aktionen manipuliert werden sollen, benötigen auf der APEX-Seite verpflichtend eine statische ID, ansonsten werden sie vom Plugin nicht erkannt. Schaltflächen können darüber hinaus auch in Regelbedingungen verwendet werden, Regionen (naturgemäß) nicht

Seitenelemente werden in der Tabelle SCT\_PAGE\_ITEM gespeichert.

#### Seitenelementtyp, Tabelle SCT\_PAGE\_ITEM\_TYPE

Seitenelemente können vom Typ ELEMENT, BUTTON oder REGION sein. Diese Seitenelementtypen werden in einer eigenen Tabelle, SCT\_PAGE\_ITEM\_TYPE gespeichert. Zusätzlich zur technische, alphanumerischen ID und dem Klartextnamen der Typen werden für Seitenelementtypen hinterlegt, ob diese Typen in APEX einen Session State besitzen und, ob und wenn ja, welchen Event diese Elementtypen binden sollen. Für die vorgegebenen Elementtypen ist festgelegt, dass nur die Typen ELEMENT und APP\_ELEMENT einen Session State besitzen und die Elementtypen ELEMENT, APP\_ELEMENT und BUTTON die Events CHANGE bzw. CLICK binden. Regionen können daher weder in Regelbedingungen verwendet werden, noch können sie einen Session State besitzen. Regionen können aber manipuliert (z.B. sichtbar oder unsichtbar, eingeklappt oder ausgeklappt oder aber aktualisiert) werden.

### Verarbeitung einer Regel

Grundlage des Plugins ist die Überführung von Entscheidungslogik in eine SQL-Abfrage. Dies soll an einem Beispiel verdeutlicht werden.

#### Verarbeitungsbeispiel

Eine APEX-Anwendungsseite enthält die beiden Elemente P1\_VALID\_FROM und P1\_VALID\_TIL. Es soll festgelegt werden, dass wenn ...

* P1\_VALID\_FROM den Wert NULL enthält, Feld P1\_VALID\_TIL deaktiviert werden soll
* P1\_VALID\_FROM größer oder gleich P1\_VALID\_TIL ist, eine Fehlermeldung bei Feld P1\_VALID\_TIL ausgegeben werden soll
* P1\_VALID\_FROM einen Wert enthält, Feld P1\_VALID\_TIL aktiviert werden soll

Es ist sichergestellt, dass eine Regelgruppe für die APEX-Anwendungsseite angelegt wurde, dass Aktionstypen zum Aktivieren und Deaktivieren von Anwendungselementen bereitstehen (die Aktionstypen implementieren die JavaScript-Funktionen apex.item('#ITEM#').disable bzw. enable).

Schließlich sind drei Einzelregeln angelegt worden:

* Bedingung mit ID 5: P1\_VALID\_FROM is NULL  
  Aktion: P1\_VALID\_TIL: Deaktivieren
* Bedingung mit ID 6: P1\_VALID\_FROM >= P1\_VALID\_TIL  
  Aktion: P1\_VALID\_TIL: Zeige Fehlermeldung
* Bedingung mit ID 7: P1\_VALID\_FROM is not NULL  
  Aktion: P1\_VALID\_TIL: Aktivieren

Die Reihenfolge der Regeln ist hier von Bedeutung: Die Regel mit der ID 7 muss nach Regel 6 ausgeführt werden, da ansonsten Regel 6 nicht mehr berücksichtigt würde. Die Bedingungen müssen also vom Speziellen zum Allgemeinen angelegt sein. Typischerweise werden zunächst Initialisierungsregeln formuliert, anschließend Fehlerstatus und zuletzt das Verhalten im Normalfall.

Aus der Analyse der Einzelbedingungen ergibt sich, dass die Elemente P1\_VALID\_FROM und P1\_VALID\_TIL referenziert wurden. Daher ist es für eine Analyse der Regeln mit Hilfe von SQL wichtig, den aktuellen Wert dieser beiden Elemente zu kennen. Einzige Quelle für SQL-Anweisungen, um den Wert von Anwendungselementen zu erfragen, ist der Session State. Da dieser aber nicht notwendigerweise synchron zur Anwendungsseite ist, wird zunächst sichergestellt, dass die aktuellen Elementwerte der Anwendungsseite in den Session State kopiert werden.

Um aus einer Regel die relevanten Elemente zu filtern, wird ein regulärer Ausdruck benutzt, der alle in Frage kommenden Elemente (alle Elemente, die auf der APEX-Seite verfügbar sind, die Anwendungselemente und verfügbar gemachte Schaltflächen) in den Regelbedingungen sucht.

Anschließend kann mit folgender SQL-Anfrage der Wert der Anwendungselemente erfragt werden:

select to\_date(v('P1\_VALID\_FROM'), 'dd.mm.yyyy') valid\_from,  
 to\_date(v(P1\_VALID\_TIL'), 'dd.mm.yyyy') valid\_til  
 from dual;

Das Plugin erstellt diese Abfrage automatisch, basierend auf den Metadaten aus den Tabellen SCT\_PAGE\_ITEM und SCT\_RULE. Zusätzlich nimmt die Abfrage noch weitere Informationen auf:

* Das auslösende Element der Anfrage wird durch die Spalte FIRING\_ITEM zugänglich gemacht und liefert die ID des Elements. Wird das Plugin initialisiert, enthält diese Spalte den Wert DOCUMENT, ansonsten den Namen des auslösenden Elements oder die statische ID der auslösenden Schaltfläche, jeweils in Versalien.
* Ist das auslösende Element DOCUMENT, wird das Plugin also initialisiert, enthält die Spalte INITIALIZING den Wert 1, ansonsten 0.
* Die Elemente, die in einer Regel angesprochen werden, steht über die Spalte FIRING\_ITEMS in der Regelview zur Verfügung und werden verwendet, um nur die Regeln zu evaluieren, die eine Abhängigkeit zum auslösenden Element besitzen. Hierfür wird das auslösende Element (Spalte FIRING\_ITEM) mit INSTR() gegen die Spalte FIRING\_ITEMS gefiltert. Zusätzlich zu den angesprochenen Elementen enthält die Spalte FIRING\_ITEMS noch den Eintrag DOCUMENT, so dass beim Initialisieren der Seite alle Regeln evaluiert werden, da in diesem Fall FIRING\_ITEM den Wert DOCUMENT enthält.

Die zusätzlichen Spalten dienen der vereinfachten Formulierung von Regeln. So kann eine Regel, die die Seitenelemente P1\_VALID\_FROM und P1\_VALID\_TIL vergleicht, durch eine zusätzliche Referenz auf die Spalte FIRING\_ITEM so eingegrenzt werden, dass sie nur angewendet wird, wenn das Element P1\_VALID\_TIL sich ändert:

P1\_VALID\_FROM >= P1\_VALID\_TIL and FIRING\_ITEM = 'P1\_VALID\_TIL'.

Die Spalte INITIALIZING kann verwendet werden, um Regeln zu erstellen, die nur bei der Initialisierung der Anwendungsseite ausgeführt werden: INITIALIZING = 1.

Schaltflächen werden ebenfalls durch Spalten zugänglich gemacht. Existiert z.B. eine Schaltfläche mit der statischen ID B1\_SAVE, würde eine Spalte B1\_SAVE enthalten sein, die immer dann, wenn B1\_SAVE auslösendes Element war, den Wert 1 enthält, ansonsten 0. Dadurch kann eine einfache Regel erstellt werden, die nur dann angewendet wird, wenn Schaltfläche B1\_SAVE auslösendes Element war: B1\_SAVE = 1. Alternativ könnte die gleiche Bedingung mit Hilfe der Spalte FIRING\_ITEM formuliert werden: FIRING\_ITEM = 'B1\_SAVE'.

Die Daten des Session State stehen in einer Inner View mit dem Namen SESSION\_STATE zur Verfügung, die im Anschluss durch die Bedingungen gefiltert wird. Die Regelbedingungen sind aufgebaut wie folgt: ((SRU\_ID = <Regel-ID>) and (<Regelbedingung>)). Alle Einzelregeln werden durch eine OR-Verbindung in der WHERE-Klausel zusammengefasst. Daraus ergibt sich für unsere Beispielregelgruppe folgende WHERE-Bedingung:

with SESSION\_STATE as (...)  
select ...  
 from SESSION\_STATE  
 join ...  
 WHERE ((sru\_id = 5) and (P1\_VALID\_FROM is NULL))  
 or ((sru\_id = 6) and (P1\_VALID\_FROM >= P1\_VALID\_TIL))  
 or ((sru\_id = 7) and (P1\_VALID\_FROM is not NULL))

Je nach Zustand des Session State werden unterschiedliche Regeln ausgewählt, deren Bedingungen aktuell zu TRUE evaluieren. Die Abfrage wird, basierend auf dem Sortierkriterium der Einzelregeln (Spalte SORT\_SEQ der Tabelle SCT\_RULE) die Einzelregel mit der niedrigsten Sortierung auswählen und anwenden. Basierend auf der ausgewählten Einzelregel stehen dann die Aktionen in PL/SQL und JavaScript über die Verbindung der Tabellen SCT\_RULE, SCT\_RULE\_ACTION und SCT\_ACTION\_TYPE zur Verfügung und können durch den Code zu einer Antwort zusammengefasst werden.

#### Rekursive Regelausführung

Wird in einer Einzelregel der Session State eines Elements verändert, wird dies, wie bereits beschrieben, im Plugin registriert, wenn die Methode PLUGIN\_SCT.SET\_SESSION\_STATE verwendet wird. Dies hat zunächst zur Folge, dass der geänderte Wert an die Anwendung zurückgemeldet und dort in das Seitenelement aktualisiert wird.

Zu diesem Verhalten existiert eine Erweiterung, die immer dann Berücksichtigung findet, wenn das Element

* nicht identisch mit dem auslösenden Element ist und
* das Element ein »relevantes« Element ist, sich also Einzelregeln der Regelgruppe auf dieses Element beziehen.

In diesem Fall wird das Plugin *nach* der Bearbeitung des ursprünglichen Events, aber *vor* der Erstellung einer Antwort, für jedes in Frage kommende Element ein CHANGE-Ereignis simulieren und eine entsprechende Antwort des Plugin erfragen. Durch diesen Mechanismus ist es möglich, in einer Aktion mehrere Elemente mit Werten zu belegen und - vor der Antwort an das aufrufende Programm - alle geänderten Elemente, auf die sich eine Regel bezieht, zu prüfen, um festzustellen, ob diese Elemente z.B. angezeigt werden sollen oder nicht. Alle Datenbankaktionen, die sich durch diese Prüfungen ergeben, werden ausgeführt, alle JavaScript-Aktionen in einer Antwort gesammelt. Zur Unterscheidung, welche JavaScript-Anweisung durch welche Regel erzeugt wurde, enthält die Antwort einen Kommentareintrag vor jedem erneuten Aufruf mit Angabe zur Regel und zur Rekursionstiefe, die diese JavaScript-Anweisung erzeugt hat.

Da durch diesen Mechanismus Endlosschleifen auftreten können, wird im Plugin ein Rekursionszähler aktualisiert, der bei maximal 10 Rekursionen die weitere Verarbeitung abbricht. Die Rekursion wird für ein Element, das bereits bearbeitet wurde, nicht erneut aufgerufen, egal, auf welcher Rekursionstiefe dieser erneute Aufruf geschieht.

Als Anwendungsbeispiel für diese rekursive Regelausführung nehmen wir folgendes an:

Ein Eingabefeld auf dem Formular erlaubt die Eingabe einer Kunden-ID. Durch Änderung der Kunden-ID soll das Plugin mehrere Felder des Kunden mit Daten füllen. Ist der Kunde ein gewerblicher Kunde, soll das Feld P1\_FIRMA angezeigt werden, ansonsten soll dieses Element verborgen werden.

Eine Einzelregel wird für das Feld P1\_KUNDE\_ID angelegt, die besagt, dass sie ausgeführt werden soll, wenn das Feld nicht leer ist. Die Aktion soll darin bestehen, eine PL/SQL-Methode aufzurufen, die mehrere Elemente im Session State mit Werten dieses Kunden belegt. Zwei weitere Einzelregeln besagen, dass, wenn das Feld P1\_KUNDE\_ART den Wert G enthält, das Feld P1\_FIRMA gezeigt werden soll, ansonsten nicht.

Wird nun das Feld P1\_KUNDE\_ART durch die PL/SQL-Methode gesetzt, würde ohne rekursive Behandlung der Regeln der CHANGE-Event für das Element nicht ausgelöst und die verbundene Regel nicht berücksichtigt. Durch die rekursive Regelausführung erkennt das Plugin, dass das Feld P1\_KUNDE\_ART geändert wurde und das sich eine Einzelregel auf dieses Feld bezieht. Dadurch wird die Regelverarbeitung für dieses Element als auslösendem Element erneut ausgeführt und die JavaScript-Aktion der Antwort hinzugefügt. Da durch diese Einzelregel keine weiteren Elemente geändert wurden, wird keine weitere rekursive Aktivität ausgelöst und die Antwort an die aufrufende Seite gesendet.

Das folgende Listing zeigt eine typische Antwort für eine relativ tief verschachtelte Seite. Hier ist das Szenario, dass nacheinander fünf Auswahllisten ausgefüllt werden müssen, um eine Regelgruppe in eine andere Anwendung zu kopieren. Zu Beginn sollen, wenn die oberste Auswahlliste nicht belegt ist, alle anderen Auswahllisten deaktiviert werden. Ohne Rekursion müssten die Regeln für die erste Auswahlliste den Zustand aller weiterer Auswahllisten definieren, ebenso die folgende für alle weiteren etc. Mit Rekursion ist das nicht nötig: Die erste Auswahlliste deaktiviert die zweite, die wiederum die dritte etc. Mittels Rekursion ergibt sich folgende (formatierte) Antwort:

<script>

de.condes.plugin.sct.setItemValues(

[{"id":"P4\_SGR\_PAGE\_ID","value":""},

{"id":"P4\_SGR\_ID","value":""},

{"id":"P4\_SGR\_APP\_TO","value":""},

{"id":"P4\_SGR\_PAGE\_TO","value":""}]);

de.condes.plugin.sct.setErrors(

{ "count":0,

"errorDependentButtons":"B4\_COPY",

"firingItems":"P4\_SGR\_PAGE\_ID,P4\_SGR\_ID,P4\_SGR\_APP\_TO,P4\_SGR\_PAGE\_TO",

"errors":[]});

//Recursion 1: RULE\_10 (Quellanwendung ist leer), Firing Item: DOCUMENT

apex.item('P4\_SGR\_PAGE\_ID').show;

apex.item('P4\_SGR\_PAGE\_ID').disable();

//Recursion 2: RULE\_30 (Quellseite ist leer), Firing Item: P4\_SGR\_PAGE\_ID

apex.item('P4\_SGR\_ID').show;

apex.item('P4\_SGR\_ID').disable();

//Recursion 3: RULE\_50 (Regelgruppe ist leer), Firing Item: P4\_SGR\_ID

apex.item('P4\_SGR\_APP\_TO').show;

apex.item('P4\_SGR\_APP\_TO').disable();

//Recursion 4: RULE\_70 (Zielanwendung ist leer), Firing Item: P4\_SGR\_APP\_TO

apex.item('P4\_SGR\_PAGE\_TO').show;

apex.item('P4\_SGR\_PAGE\_TO').disable();

//Recursion 5: RULE\_90 (Zielseite ist leer), Firing Item: P4\_SGR\_PAGE\_TO

apex.item('B4\_COPY').show();

apex.item('B4\_COPY').disable();

</script>

Zur Vermeidung schwer wartbaren Codes sollte darauf geachtet werden, Rekursionen nicht zu komplex werden zu lassen, wo immer möglich. Der Sinn des Plugins besteht in der Vereinfachung von Code, nicht in dessen Gegenteil.

#### Technische Umsetzung

Im Kern ist SCT\_ADMIN ein Codegenerator, der, basierend auf den Metadaten der Regelgruppen, SQL-Abfragen oder Funktionsaufrufe erzeugt. Dabei werden SQL-Templates, die im Package SCT\_CONS für die verschiedenen Anwendungsfälle vorgehalten werden, in Schleifen über die Metadaten mit Daten aus den Regelgruppen angereichert und entweder ausgeführt oder an die aufrufende Umgebung weitergereicht.

Derzeit ist folgendes Verhalten implementiert:

* Wird eine Regelgruppe oder Einzelregel angelegt oder verändert, wird eine SQL-Abfrage erstellt, wie im Beispiel erläutert, und als Regelview in der Datenbank abgelegt.
* Wird eine Aktion erstellt, wird die angelegt Regelview gegen den Session State ausgeführt, die resultierende Einzelregel ermittelt und aus den Metadaten dieser Einzelregel die Antwort generiert und an den aufrufenden Code übermittelt
* Wird eine Regelgruppe exportiert, werden die Metadaten in Aufrufe der Administrationspackages überführt und diese als CLOB an den aufrufenden Code übermittelt

Die Logik des Plugins liegt in der sinnvollen Formulierung der Metadaten und der Aktionstypen, nicht im Plugin selbst. Beispielhaft soll dies an der (leicht verkürzten) Methode CREATE\_RULE\_VIEW verdeutlicht werden, die der Erzeugung der Regelview für eine Regelgruppe dient:

procedure CREATE\_RULE\_VIEW(

p\_sgr\_id in sct\_group.sgr\_id%type)

as

l\_view\_name varchar2(30 byte);

l\_data\_cols varchar2(32767);

l\_where\_clause varchar2(32767);

l\_stmt varchar2(32767);

begin

l\_view\_name := SCT\_CONSt.c\_view\_name\_prefix || p\_sgr\_id;

l\_data\_cols := CREATE\_COLUMN\_LIST(p\_sgr\_id);

l\_where\_clause := CREATE\_WHERE\_CLAUSE(p\_sgr\_id);

-- Erzeuge die Regelview

l\_stmt := utl\_text.bulk\_replace(SCT\_CONSt.c\_create\_view\_template

|| SCT\_CONSt.c\_rule\_view\_template, char\_table(

'#NAME#', l\_view\_name,

'#DATA\_COLS#', l\_data\_cols,

'#WHERE\_CLAUSE#', l\_where\_clause,

'#SGR\_ID#', p\_sgr\_id));

execute immediate l\_stmt;

end CREATE\_RULE\_VIEW;

Die Methoden CREATE\_COLUMN\_LIST bzw. CREATE\_WHERE\_CLAUSE arbeiten analog und liefern die Inner View SESSION\_STATE bzw. die WHERE-Klausel.

Methoden, die auf ähnliche Weise Code generieren, sind:

* CREATE\_RULE\_VIEW
* CREATE\_COLUMN\_LIST
* CREATE\_WHERE\_CLAUSE
* VALIDATE\_RULE (mit der Erweiterung, dass der Code direkt ausgeführt und Kompilierfehler angezeigt werden)
* EXPORT\_RULE\_GROUP, EXPORT\_RULE\_GROUPS
* CREATE\_ACTION

## Stammdatenpflege

Sie Metadaten des Plugins werden einerseits durch Assistenten-basierte Seiten der APEX-Anwendung direkt gepflegt, andererseits existiert eine API zur Erstellung von Regeln, zum Beispiel aus einem Export der Regelgruppe oder der Regelgruppen. Zudem bietet das Package Hilfsmethoden zum Löschen von Regelgruppen oder zur Resequenzierung der Spalten SORT\_SEQ an. Die hierfür vorhandenen Methoden des Packages stellen einfache Wrapper um MERGE-Anweisungen dar, mit deren Hilfe die Stammdatentabellen gepflegt werden können.

Methoden, die auf diese Weise arbeiten, sind:

* MERGE\_RULE\_GROUP
* MERGE\_RULE
* MERGE\_RULE\_ACTION
* MERGE\_ACTION\_TYPE
* DELETE\_RULE\_GROUP
* RESEQUENCE\_RULE\_GROUP

## Hilfsmethoden