

# White Paper

## Use-Cases zur Fachanwendungsintegration

Fabasoft Folio 2026

Copyright © Fabasoft R&D GmbH, A-4020 Linz, 2026. Alle Rechte vorbehalten. Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Marken der jeweiligen Hersteller.

Durch die Übermittlung und Präsentation dieser Unterlagen alleine werden keine Rechte an unserer Software, an unseren Dienstleistungen und Dienstleistungsergebnissen oder sonstigen geschützten Rechten begründet.

Aus Gründen der einfacheren Lesbarkeit wird auf die geschlechtsspezifische Differenzierung, z. B. Benutzer/-innen, verzichtet. Entsprechende Begriffe gelten im Sinne der Gleichbehandlung grundsätzlich für beide Geschlechter.

## Inhalt

1 Einleitung .....	5
2 Softwarevoraussetzungen.....	5
3 Fachanwendungsintegration mit einer Fabasoft Produktinstallation.....	5
4 Architektur einer Fachanwendung .....	6
4.1 Datenhaltung.....	6
4.2 Kommunikation.....	7
4.3 Workflow .....	8
5 Softwarekomponenten.....	8
6 Objektklassen der Fachanwendungsintegration .....	9
6.1 Fachanwendungsdefinition.....	9
6.2 Fachanwendungsbereichsdefinition.....	9
6.2.1 Schema für Fachanwendungsdaten .....	10
6.2.2 Namespaces .....	10
7 Objektmodellerweiterung der Fachanwendungsintegration.....	10
7.1 Fachanwendungsdaten .....	11
7.2 Fachanwendungs-Datensatz .....	11
7.3 Verfügbare Fachanwendungsbereiche .....	12
7.4 Daten des Fachobjekts .....	12
8 Funktionalität der Fachanwendungsintegration.....	12
8.1 Initialisieren eines neuen Datensatzes im Fachanwendungsaggregat .....	12
8.2 Zugriff auf Fachanwendungsdaten .....	13
8.3 Anzeige der Fachanwendungsdaten.....	14
8.4 Fachanwendungsaufrufe über Kontextmenübefehl.....	14
8.5 Suche .....	15
9 Authentifizierung.....	15
9.1 Active-Directory-Umgebung.....	17
9.2 Benutzerkonto für die Delegation konfigurieren.....	17
9.3 Benutzerkonto, unter dem die Fabasoft Folio Webservices laufen, konfigurieren .....	17
10 SOAP-Aktionen der Fabasoft Produktinstallation.....	18
10.1 Technische Aspekte der SOAP-Schnittstelle.....	18
10.2 Aufbau des Service-URL.....	19
10.3 Webservice-Definition.....	20

11 Funktionsteilbereich „Aufruf von Funktionalität der Fabasoft Produktinstallation durch eine Fachanwendung“ .....	22
11.1 Lösungsbereich Transaktionen .....	22
11.1.1 Mehrere SOAP-Aktionen zu einer Transaktion zusammenfassen .....	22
11.2 Lösungsbereich Fachanwendungsdaten .....	22
11.2.1 XML-Fachanwendungsdaten schreiben .....	22
11.2.2 XML-Fachanwendungsdaten lesen .....	23
11.2.3 Fachdokument in der Fabasoft Produktinstallation anzeigen .....	24
11.2.4 Fachanwendungsdaten in Dokument-Eigenschaften verwenden .....	24
11.3 Lösungsbereich Objekt-Bearbeitung .....	24
11.3.1 Objekt bearbeiten .....	25
11.3.2 DMS-Dokument erzeugen .....	26
11.3.3 Inhalt eines DMS-Objekts bearbeiten .....	27
11.3.4 Versionierung .....	29
11.3.5 Neues Objekt erzeugen .....	33
11.4 Lösungsbereich Workflow .....	35
11.4.1 Neue Prozessinstanz erzeugen .....	35
11.4.2 Erstellen von Aktivitäten .....	36
11.4.3 Einfügen von Aktivitäten .....	37
11.4.4 Vorschreiben (FSCBAIWF@1.1001:SOAPPrescribeObject) .....	38
11.4.5 Zuteilen .....	39
11.4.6 Weiterleiten/Beenden einer Aktivität (FSCBAIWF@1.1001:SOAPSetCompleted) ....	41
11.4.7 Unterschreiben .....	41
11.5 Lösungsbereich Suche .....	42
11.5.1 Objekte via XML/SOAP suchen und Eigenschaften auslesen .....	42
11.6 Lösungsbereich Check-out - Check-in .....	44
11.6.1 Auschecken über SOAP (FSCBAI@1.1001:SOAPCheckOut) .....	44
11.6.2 Einchecken über SOAP (FSCBAI@1.1001:SOAPCheckIn) .....	45
11.7 Generische SOAP-Aktionen .....	47
11.7.1 Generisches Lesen von Eigenschaften von Objekten .....	47
11.7.2 Generisches Ändern von Eigenschaften von Objekten .....	50
12 Funktionsteilbereich „Aufruf von SOAP-Aktionen einer Fachanwendung durch die Fabasoft Produktinstallation“ .....	53
12.1 Lösungsbereich „Nutzung von Funktionalität einer Fachanwendung“ .....	53

12.1.1 Standardinitialisierung eines Objekts über SOAP .....	53
12.1.2 Aufruf einer SOAP-Aktion mit XML-Parametern.....	55
12.1.3 Senden einer einfachen SOAP-Nachricht.....	56
12.1.4 Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich .....	57
12.1.5 Laden des Objekts aus einem CONTENT mit XML-Daten .....	58
12.1.6 Ausgeben des Objekts in einen CONTENT mit XML-Daten .....	58
12.1.7 Kombiniertes SOAP-Client-/GUI-Aufruf .....	59
<b>13 Funktionsteilbereich „Konfiguration“ .....</b>	<b>60</b>
13.1 Konfiguration für Fachanwendungen .....	60
13.1.1 URL für lokales WebDAV .....	60
13.1.2 Basis-URLs.....	60
13.1.3 Fachanwendungsaufrufe.....	62
13.1.4 Menü für Fachdokumente .....	65
13.1.5 Transformation für Fachdokumente .....	66
13.1.6 Weiterführen von permanenten Sperren .....	67
13.1.7 Schematazuordnungen.....	68
<b>14 Glossar .....</b>	<b>69</b>

# 1 Einleitung

Dieses Dokument gibt einen Überblick über die Basis-**Webservices** einer Fabasoft Produktinstallation, die zur Integration von Fachanwendungen genutzt werden können.

Die Nutzung von **Use-Cases** sowohl auf Ebene der grafischen Benutzeroberfläche (GUI) als auch – soweit in der Fabasoft Produktinstallation verfügbar – im Wege der zugehörigen Use-Case-basierten Webservice-Schnittstellen der Fabasoft Produktinstallation ist für registrierte Benutzer möglich.

Weiterhin werden in diesem Dokument alle verfügbaren SOAP-Aktionen und technischen Raffinessen der Fachanwendungsintegration beschrieben und anhand von Beispielen näher erläutert.

## 2 Softwarevoraussetzungen

**Systemumgebung:** Die Informationen in diesem Dokument beziehen sich auf eine Microsoft Windows-Systemumgebung.

**Unterstützte Plattformen:** Detaillierte Informationen zu unterstützten Betriebssystemen und unterstützter Software finden Sie in der Softwareproduktinformation auf dem Distributionsmedium.

## 3 Fachanwendungsintegration mit einer Fabasoft Produktinstallation

Die Integration einer Fachanwendung in einer Fabasoft Produktinstallation erfolgt unter folgenden Zielsetzungen:

Qualität für den Anwender durch

- Hochverfügbarkeit
- Stabilität
- Performance

Qualität für den Administrator durch

- Skalierbarkeit
- Sicherheit
- Konsistenz
- Überschaubarkeit der Integration

Eine Fachanwendung im Sinne einer Fabasoft Produktinstallation ist unterteilt in *Fachanwendungsbereiche* („FA-Bereiche“). Ein *Fachanwendungsbereich* bildet eine eindeutig abgegrenzte Einheit für den Zugriff auf die Fachanwendung und die Fachanwendungsdaten. Eine Fachanwendung besteht aus mindestens einem *Fachanwendungsbereich*, in dessen Kontext Konfigurationseinstellungen vorgenommen werden können.

## 4 Architektur einer Fachanwendung

Die Integration von Fachanwendungen erfolgt nach dem Prinzip der losen Kopplung um die Einflüsse der Fachanwendung auf den Betrieb der Fabasoft Produktinstallation möglichst gering zu halten. Die **Stammdomäne** der Fabasoft Produktinstallation und die Fachanwendung sind über Webservices verbunden. Das ermöglicht eine GUI-, Daten- und Workflowkopplung zwischen der Fabasoft Produktinstallation Installation und der Fachanwendung gemäß der in diesem Dokument definierten Use-Cases. Eine darüber hinausgehende Integration wird in produktspezifischen White Papers beschrieben, die sich auf der jeweiligen Produkt-DVD befinden.

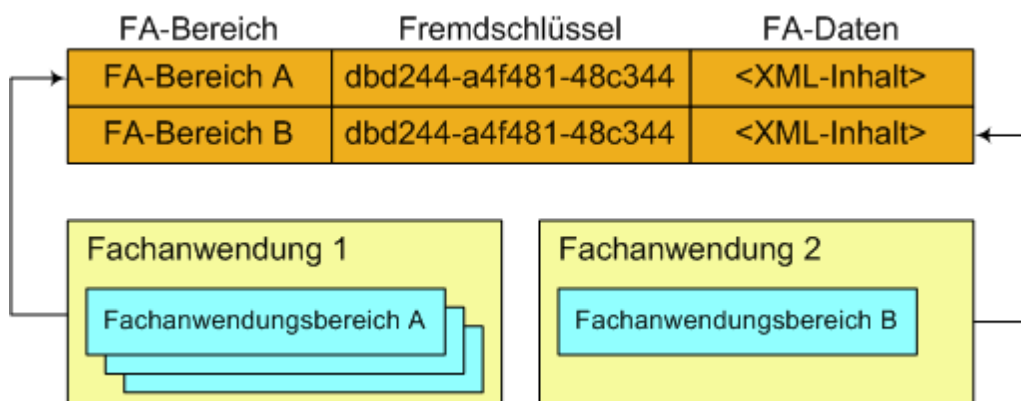
### 4.1 Datenhaltung

Alternativ bzw. zusätzlich zur Datenspeicherung in der **Fachanwendungsdomäne** kann eine Fachanwendung ihre Daten auch bei den Geschäftsobjekten in der Fabasoft Produktinstallation speichern. Diese Speicherung der Daten in der Fabasoft Produktinstallation hat folgende Vorteile:

- In der Fabasoft Produktinstallation kann auf die Daten zugegriffen werden, ohne eine Anfrage an die Fachanwendung durchführen zu müssen.
- Die Daten können über eine **XSL**-Transformation angezeigt und in Microsoft Office-Dokumenten (mittels Dokument-Eigenschaften) verwendet werden.
- Beim Geschäftsobjekt in der Fabasoft Produktinstallation sind alle definierten Daten zum Geschäftsfall vorhanden.
- Die Sicherung der Daten der Fachanwendung kann an zentraler Stelle erfolgen.
- Verwendung der Fabasoft Referenzarchitektur

Die *Fachanwendungsdaten* werden dabei in Form einer **XML**-Struktur gespeichert. Dazu existiert in der Fabasoft Produktinstallation zu jedem Geschäftsobjekt eine *Aggregatsliste* mit den entsprechenden Eigenschaften.

Skizzierte Darstellung:



Der XML-Inhalt der *Fachanwendungsdaten* in der Fabasoft Produktinstallation wird mittels Aufruf der entsprechenden SOAP-Aktionen der Fabasoft Produktinstallation von der Fachanwendung gelesen bzw. geschrieben.

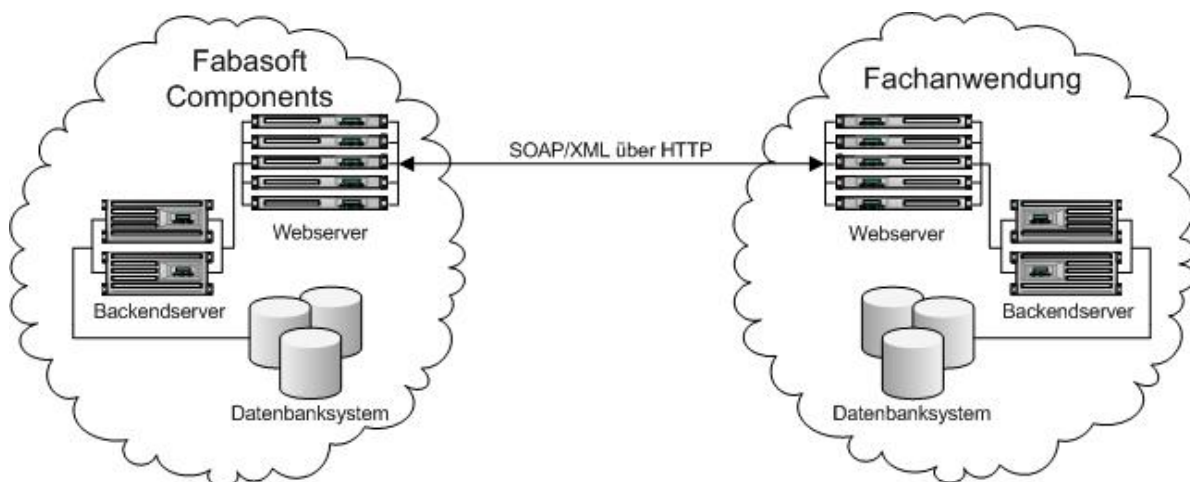
Es können zu einem Geschäftsobjekt auch mehrere *Fachanwendungsdaten* gespeichert werden. Diese gespeicherten Daten werden anhand des *Fachanwendungsbereichs* unterschieden. Pro *Fachanwendungsbereich* kann genau ein XML-Inhalt in einem Geschäftsobjekt gespeichert werden. Durch die Speicherung des Fremdschlüssels (Eindeutiger Schlüssel des Objekts in der Fachanwendung) kann das entsprechende Objekt in der Fachanwendung schnell gefunden werden. Dieser generische Datenzeiger muss von der Fachanwendung erzeugt werden.

## 4.2 Kommunikation

Die Fachanwendung erzeugt und bearbeitet eigene Daten. Daneben hat die Fachanwendung die Möglichkeit, Abläufe in der Fabasoft Produktinstallation zu nutzen, z. B. Funktionalitäten des Workflows. In Verbindung mit der Workflow-Komponente der Fabasoft Produktinstallation und der Fachanwendungsintegration können anwendungsübergreifende Use-Cases verwirklicht werden:

Fachanwendungen können Aktionen aufrufen, um z. B. Objekte in der Fabasoft Produktinstallation zu erzeugen. Dazu bedient sich eine Fachanwendung der in diesem Dokument beschriebenen Use-Cases.

Falls die Daten der Fachanwendung außerhalb der Fabasoft Produktinstallation Domäne gespeichert werden, muss durch einen generischen Datenzeiger (ein eindeutiger Schlüssel) eine Verknüpfung zwischen Objekten in der Fabasoft Produktinstallation und den Daten der Fachanwendung hergestellt werden.



Damit die Fabasoft Produktinstallation auch Funktionalitäten der Fachanwendung nutzen kann, ist es möglich, diese aus der Fabasoft Produktinstallation heraus entsprechend der Beschreibung im Kapitel 12 „Funktionsteilbereich „Aufruf von SOAP-Aktionen einer Fachanwendung durch die Fabasoft Produktinstallation“ aufzurufen. Da der Zugriff auf Daten der Fachanwendung über Webservices durchgeführt wird, ist es erforderlich, dass die Fachanwendung entsprechende SOAP-Services für den Datenzugriff zur Verfügung stellt.

Übergreifende verteilte **Transaktionen** zwischen der Fabasoft Produktinstallation Installation und der Fachanwendung werden nicht unterstützt, da das nicht dem Prinzip einer losen Kopplung entsprechen würde.

Die Datenkopplung von Fachanwendungen und Fabasoft Produktinstallationen erfolgt über SOAP und WSDL. SOAP ist ein XML-basiertes Protokoll für den Austausch von Informationen in einer dezentralen, verteilten Umgebung. Die SOAP-Implementierung der Fabasoft Produktinstallation unterstützt ausschließlich das **HTTP**-Protokoll.

Die Beschreibung der zur Verfügung gestellten Webservices erfolgt über WSDL (Web Service Description Language).

Die Verwendung der Webservices der Fabasoft Produktinstallation entspricht einem Request-Response-Modell. Um die Webservices anzusprechen, wird die „Document-Style“-Aufrufkonvention des SOAP-Protokolls verwendet. Das Webservice erhält über den SOAP-Request ein XML-Dokument als Input. Der SOAP-Response, der dem Aufruf folgt, enthält wiederum ein XML-Dokument, das die Output-Daten des Webservice beschreibt.

### 4.3 Workflow

In der Fabasoft Produktinstallation besteht die Möglichkeit Geschäftsprozesse auf folgende Arten abzubilden:

- Vordefinierte („strukturierte“) Geschäftsprozesse
- Ad-hoc definierte Geschäftsprozesse
- Kombination aus vordefinierten und ad-hoc definierten Geschäftsprozessen

Die Fachanwendung nutzt die vorhandene Workflow-Funktionalität der Fabasoft Produktinstallation für alle Workflow-relevanten Schritte. Den Fachanwendungen stehen zur Steuerung des Workflows SOAP-Aktionen der Fabasoft Produktinstallation zur Verfügung.

## 5 Softwarekomponenten

Folgende Softwarekomponenten der Fabasoft Produktinstallation werden (teils optional) für die Fachanwendungsintegration benötigt:

---

Fabasoft Folio/Base		
COOXML@1.1	<i>Integration for XML</i>	Basiskomponente für XML; enthält die <b>Objektklassen</b> für XML-Mappings und XML-Schemata.
FSCOWS@1.1001	<i>Open Web Services</i>	Komponente für Webservices; enthält die Objektklassen für <i>HTTP-Anschlüsse</i> und <i>Webservice-Definitionen</i> .
FSCBAI@1.1001	<i>Integration for Business Application</i>	Grundlage um externe Fachanwendungen verwenden zu können; muss nicht installiert werden, wenn die Domäne selbst nur als Fachanwendung verwendet wird.

---

FSCBA@1.1001	<i>Business Application Extension</i>	<p>Erweitert die Objektklasse <i>Objekt</i> (COOSYSTEM@1.1:Object) um Möglichkeiten, jedes Objekt als Fachobjekt zu betrachten. Das Fachobjekt kann von einer Fabasoft Folio Domäne über SOAP initialisiert werden.</p> <p>Die Installation dieser Softwarekomponente ist nur erforderlich, wenn die Fachanwendung in Fabasoft Folio entwickelt werden soll.</p>
--------------	---------------------------------------	--

---

## Fabasoft Folio/WF

FSCBAIWF@1.1001	<i>Integration for Business Application: Workflow</i>	<p>Stellt SOAP-Aktionen zur Verfügung, die eine Steuerung des Workflows ermöglichen. Das ist nur erforderlich, wenn die Fachanwendung diese Funktionalität nutzt.</p>
-----------------	---	---

---

## Fabasoft Folio/WD

FSCVBAI@1.1001	<i>Integration for Business Application (Virtual Application)</i>	<p>Ermöglicht die Verwendung von Fachanwendungen im Fabasoft Folio Webclient.</p>
----------------	---	---

## 6 Objektklassen der Fachanwendungsintegration

Nach der Installation der benötigten Softwarekomponenten stehen zusätzliche Komponentenobjekte in der Fabasoft Produktinstallation zur Verfügung. Die in diesem Abschnitt angeführten Komponentenobjekte sind für die Fachanwendungsintegration von Bedeutung.

### 6.1 Fachanwendungsdefinition

Ein Objekt der Klasse *Fachanwendungsdefinition* (FSCBAI@1.1001:BusinessApp) dient der Dokumentation und ermöglicht die Zusammenfassung von *Fachanwendungsbereichen* zu einer Fachanwendung. Zu jeder Fachanwendung kann eine Beschreibung eingegeben werden. In einer Objektliste *Fachanwendungsbereiche* (FSCBAI@1.1001:depts) werden die zu dieser Fachanwendung gehörenden *Fachanwendungsbereiche* referenziert.

### 6.2 Fachanwendungsbereichsdefinition

Für die Fachanwendungsintegration wird mindestens ein Objekt der Klasse *Fachanwendungsbereichsdefinition* (FSCBAI@1.1:Department) benötigt. Eine Fachanwendung kann in beliebig viele *Fachanwendungsbereiche* unterteilt sein. Ein *Fachanwendungsbereich*

ist innerhalb einer Fachanwendung die definierte Zugriffseinheit für die *Fachanwendungsdaten* sowie der Kontext, in dem alle Konfigurationseinstellungen durchgeführt werden.

In der Implementierung der Fachanwendung werden die vollständigen *Referenzen* der *Fachanwendungsdefinitionsobjekte* verwendet. Daher ist insbesondere auf die Vergabe der *Referenzen* zu achten.

### 6.2.1 Schema für Fachanwendungsdaten

Die Fachanwendung speichert ihre Daten im XML-Format zu den Objekten, die von der Fachanwendung verwendet werden. Das Schema dieser XML-Daten gibt die Implementierung der jeweiligen Fachanwendung vor. Mithilfe der SOAP-Aktion *Setzen der Fachanwendungsdaten über XML/SOAP* (FSCBAI@1.1001:SOAPSetData) werden die Fachanwendungsdaten in Fabasoft Folio gespeichert. Bei der Speicherung kann eine Validierung gegen ein XML-Schema veranlasst werden. Um die Validierung zu aktivieren muss im *Fachanwendungsbereichsdefinitionsobjekt* in der Eigenschaft *Schema für Fachanwendungsdaten* (FSCBAI@1.1001:schemadefs) ein XML-Schema hinterlegt werden. Zusätzlich muss eine Objektklasse angegeben werden, für die dieses XML-Schema gilt. Dadurch kann ein *Fachanwendungsbereich* Objekte unterschiedlicher Klassen bedienen. Die *Fachanwendungsdaten*, die z. B. zu einem Text-Dokument gespeichert werden, können einem anderen XML-Schema entsprechen als jene, die zu einem Word-Dokument gespeichert werden.

### 6.2.2 Namespaces

Manche SOAP-Aktionen zum Lesen der Fachanwendungsdaten verwenden einen XPath-Parameter, wie z. B. *Ermitteln der Fachanwendungsdaten über XML/SOAP* (FSCBAI@1.1001:SOAPGetData), *Setzen der Fachanwendungsdaten über XML/SOAP* (FSCBAI@1.1001:SOAPSetData) und *Textelement aus den Fachanwendungsdaten ermitteln* (FSCBAI@1.1001:GetDataValue). Bei diesen Aktionen wurde bewusst auf die Übergabe eines Namespace-Aggregat-Arrays verzichtet, um bei der Verwendung der Aktionen flexibler zu sein. In den XPath-Abfragen werden jedoch XML-Namespaces in Form von Namespace-Präfixes verwendet. Die Zuordnung von Präfix zu XML-Namespace erfolgt im *Fachanwendungsbereichsdefinitionsobjekt* in der Eigenschaft *Namespaces* (FSCBAI@1.1001:namespaces). Das Präfix kann frei vergeben werden. Wichtig ist, dass bei den späteren XPath-Abfragen dasselbe Präfix (case-sensitiv) verwendet wird. Der XML-Namespace (ebenfalls case-sensitiv) wird durch die Implementierung der Fachanwendung vorgegeben. Dieser muss derselbe sein, der im beschriebenen XML-Schema-Objekt eingetragen ist.

## 7 Objektmodellerweiterung der Fachanwendungsintegration

Die beiden Softwarekomponenten *Integration for Business Application* (FSCBAI@1.1001) und *Business Application Extension* (FSCBA@1.1001) erweitern das bestehende Objektmodell um die im Folgenden beschriebenen Eigenschaften.

## 7.1 Fachanwendungsdaten

Die Softwarekomponente `FSCBAI@1.1001` erweitert die Objektklasse *Objekt* um die Aggregatsliste *Fachanwendungsdaten* (`FSCBAI@1.1001:data`). In der Fabasoft Produktinstallation existiert zu jedem Geschäftsobjekt eine solche Aggregatsliste mit folgenden Daten:

Eigenschaft	Typ	Erklärung
<i>Fachanwendungsbereich</i>	OBJECT	Objektzeiger auf ein Objekt der Klasse <i>Fachanwendungsbereichsdefinition</i> ; dieses Objekt repräsentiert einen <i>Fachanwendungsbereich</i> in der Fabasoft Folio Domäne.
<i>Fremdschlüssel</i>	STRING	Zeichenketteneigenschaft zur Speicherung des Fremdschlüssels.
<i>XML-Inhalt</i>	CONTENT	Inhalt, der ausschließlich XML-Daten enthalten darf.

Der XML-Inhalt der Fachanwendungsdaten wird mithilfe der SOAP-Aktionen `FSCBAI@1.1001:SOAPGetData` und `FSCBAI@1.1001:SOAPSetData` gelesen bzw. geschrieben. Das Aggregat *Fachanwendungsdaten* wird für die Datenhaltung von Daten der Fachanwendung benötigt. Pro *Fachanwendungsbereich* kann genau ein Datensatz in dieser Liste existieren. Weiters werden genau diese XML-Daten verwendet, um *Fachdokumente* zu erzeugen, Dokument-Eigenschaften aus *Fachanwendungsdaten* in ein Dokument einzufügen oder um über einen XPath-Ausdruck Werte (`FSCBAI@1.1001:GetDataValue`) aus diesem XML zu ermitteln.

## 7.2 Fachanwendungs-Datensatz

Die Softwarekomponente `FSCBAI@1.1001` erweitert die Objektklasse *Objekt* um das Aggregat *Fachanwendungs-Datensatz* (`FSCBAI@1.1001:rs`). Der *Fachanwendungsbereich* und der Fremdschlüssel der *Fachanwendungsdaten* werden über das Aggregat *Fachanwendungs-Datensatz* gesetzt. Der *Fachanwendungs-Datensatz* setzt sich aus den folgenden Eigenschaften zusammen:

Eigenschaft	Typ	Erklärung
<i>Fachanwendungsbereich</i>	STRING	Die vollständige <i>Referenz</i> eines <i>Fachanwendungsbereichsdefinitionsobjekts</i> ; dieses Objekt repräsentiert einen <i>Fachanwendungsbereich</i> in der Stammdomäne.
<i>Fremdschlüssel</i>	STRING	Zeichenketteneigenschaft zur Speicherung des Fremdschlüssels.

## 7.3 Verfügbare Fachanwendungsbereiche

Die Softwarekomponente `FSCBAI@1.1001` erweitert die Objektklassen *Aktuelle Domäne*, *Gruppe* und *Arbeitsumgebung* um die Aggregatsliste *Verfügbare Fachanwendungsbereiche* (`FSCBAI@1.1001:availabledept`). Diese Eigenschaft dient dazu, einen *Fachanwendungsbereich* für den jeweiligen Benutzerkreis verfügbar zu machen.

Die Aggregatsliste *Verfügbare Fachanwendungsbereiche* setzt sich aus folgenden Eigenschaften zusammen:

Eigenschaft	Typ	Erklärung
<i>Fachanwendungsbereich</i>	OBJECT	Objektzeiger auf ein Objekt der Klasse <i>Fachanwendungsbereichsdefinition</i>
<i>Softwarekomponente</i>	OBJECT	Objektzeiger auf eine <i>Softwarekomponente</i>

## 7.4 Daten des Fachobjekts

Wird eine Fachanwendung mit Fabasoft Folio umgesetzt, dann sollte in der Fachanwendungsdomäne die Softwarekomponente *Business Application Extension* (`FSCBA@1.1001`) installiert werden. Diese *Softwarekomponente* erweitert alle Objekte um das Aggregat *Daten des Fachobjekts* (`FSCBA@1.1001:data`). Dieses Aggregat setzt sich aus den folgenden Eigenschaften zusammen:

Eigenschaft	Typ	Erklärung
<i>Schlüssel der Stammanwendung</i>	STRING	Enthält den Schlüssel des zugehörigen Objekts der Stammdomäne (die Objektadresse)
<i>Fachanwendungsbereich der Stammanwendung</i>	STRING	Enthält den <i>Fachanwendungsbereich</i> als Zeichenkette

Diese Eigenschaft kann von Fachanwendungen genutzt werden, damit diese bei einem Fachanwendungsobjekt die Daten des jeweiligen Objekts in der Stammdomäne speichern kann.

# 8 Funktionalität der Fachanwendungsintegration

Die in Kapitel 5 „Softwarekomponenten“ angeführten Softwarekomponenten erweitern die Fabasoft Produktinstallation um die im Folgenden beschriebenen Funktionalitäten.

## 8.1 Initialisieren eines neuen Datensatzes im Fachanwendungsaggregat

Um einen neuen Datensatz im Aggregat *Fachanwendungsdaten* (`FSCBAI@1.1001:data`) zu erzeugen, kann das Aggregat *Fachanwendungs-Datensatz* (`FSCBAI@1.1001:rs`) mit dem

*Fachanwendungsbereich* und dem Fremdschlüssel befüllt werden. Durch die *Set-Aktion* dieses Aggregats werden die Daten in das Aggregat *Fachanwendungsdaten* eingetragen.

## 8.2 Zugriff auf Fachanwendungsdaten

Auf die Fachanwendungsdaten kann z. B. in einer Fabasoft DUCX Expression über die Aktion *Textelement aus den Fachanwendungsdaten ermitteln* (FSCBAI@1.1001:GetDataValue) zugegriffen werden. Als Parameter wird dieser Aktion ein XPath übergeben. Der Rückgabewert der Aktion liefert alle Daten ab dem im XPath angegebenen Knoten.

Die Aktion FSCBAI@1.1001:GetDataValue wird mit folgenden Parametern aufgerufen:

Parametername	IN/OUT	Typ	Erklärung
dept	In	OBJECT	Der <i>Fachanwendungsbereich</i> , für den diese Aktion aufgerufen wird
xpath	In	STRING	XPath-Ausdruck <b>Beispiel:</b> „//ns1:node1/ns1:node2“
retval	Out	STRING	Der Inhalt des im XPath bezeichneten XML-Elements wird als Ergebnis ausgegeben. Sind in diesem Element weitere Subelemente enthalten, werden auch deren Inhalte als Text ausgegeben, wobei die Knotennamen nicht berücksichtigt werden.

**Anmerkung:** Die XML-Namespaces bzw. die XML-Namespace-Präfixe werden aus dem Objekt *Fachanwendungsbereichsdefinition* aus der Eigenschaft *Namespaces* ermittelt. Es muss demnach der *Fachanwendungsbereich* übergeben werden, in dem die Namespaces definiert sind.

### Beispiel:

Verwendung in einer Fabasoft DUCX Expression zum Ermitteln der Daten für eine Dokument-Eigenschaft (vordefinierte virtuelle Eigenschaft):

```
this.FSCBAI@1.1001:GetDataValue(#SWK@1.1019:DepartmentA,  
"/ns2:data/ns2:casefile/ns2:offenses") [3])
```

Auf das aktuelle Objekt wird die Aktion FSCBAI@1.1001:GetDataValue ausgeführt. Im dritten Parameter befinden sich die Werte des über den XPath angegebenen Knotens. Der XPath sollte auf einen Blattknoten verweisen (d.h. es sollten keine weiteren Sub-Knoten enthalten sein), damit genau ein Wert ausgelesen wird.

**Wichtig:** Der XML-Namespace ns2 muss im *Fachanwendungsbereich* (in diesem Fall SWK@1.1019:DepartmentA) definiert sein.

### 8.3 Anzeige der Fachanwendungsdaten

Die *Fachanwendungsdaten* können von der Fabasoft Produktumgebung als eigene HTML-Übersicht, die über ein dynamisches Kontextmenü aufgerufen wird (siehe Kapitel 13.1.4 „Menü für Fachdokumente“) dargestellt werden.

Folgende Aktion steht dafür zur Verfügung:

*Transformierten XML-Inhalt ermitteln*  
(FSCBAI@1.1001:GetPrimaryContents)

Parametername	IN/OUT	Typ	Erklärung
trigger	In	OBJECT	Mit diesem Auslöser kann (optional) ein Kontext für das Aufrufen der Aktion angegeben werden. Das hier angegebene Objekt wird dazu verwendet, die passende Konfigurationszeile in der <i>Konfiguration für Fachanwendungen</i> (FSCBAI@1.1001:ConfigurationClass) in der Eigenschaft <i>XML Transformation für Fachdokumente</i> (FSCBAI@1.1001:xmltransformation) zu ermitteln, die dieses Objekt als Kontext eingetragen hat.
dept	In	OBJECT	Der <i>Fachanwendungsbereich</i> , für den diese Aktion aufgerufen wird.
transformed	Out	CONTENT	Der transformierte Inhalt wird als HTML-Inhalt ausgegeben.

Als Rückgabewert liefert die Aktion die transformierten *Fachanwendungsdaten* als HTML-Inhalt.

### 8.4 Fachanwendungsaufrufe über Kontextmenübefehl

Zum Starten der Fachanwendung können die Kontextmenüs von Objektklassen um den Kontextmenübefehl *Fachanwendungen* (FSCBAI@1.1001:MenuBusinessApps) erweitert werden. Abhängig davon, ob in der Eigenschaft *Fachanwendungsdaten* (FSCBAI@1.1001:data) ein Eintrag für den jeweiligen *Fachanwendungsbereich* und für die entsprechende Objektklasse vorhanden ist, stehen weitere Untermenübefehle zur Verfügung.

Durch einen Klick auf einen der Menübefehle wird die Aktion *Fachanwendung starten* (FSCBAI@1.1001:MenuStartBusinessApp) ausgeführt. Diese ruft wiederum die Aktion *Fachanwendung über Programmiername starten* (FSCBAI@1.1001:StartBusinessApp) auf.

Wird ein Kontextmenübefehl auf mehrere Objekte ausgeführt, dann wird die Aktion auf alle markierten Objekte separat angewendet.

## 8.5 Suche

Grundsätzlich muss eine Such- und Recherchefunktionalität für Daten der Fachanwendung von jeder Fachanwendung selbst zur Verfügung gestellt werden.

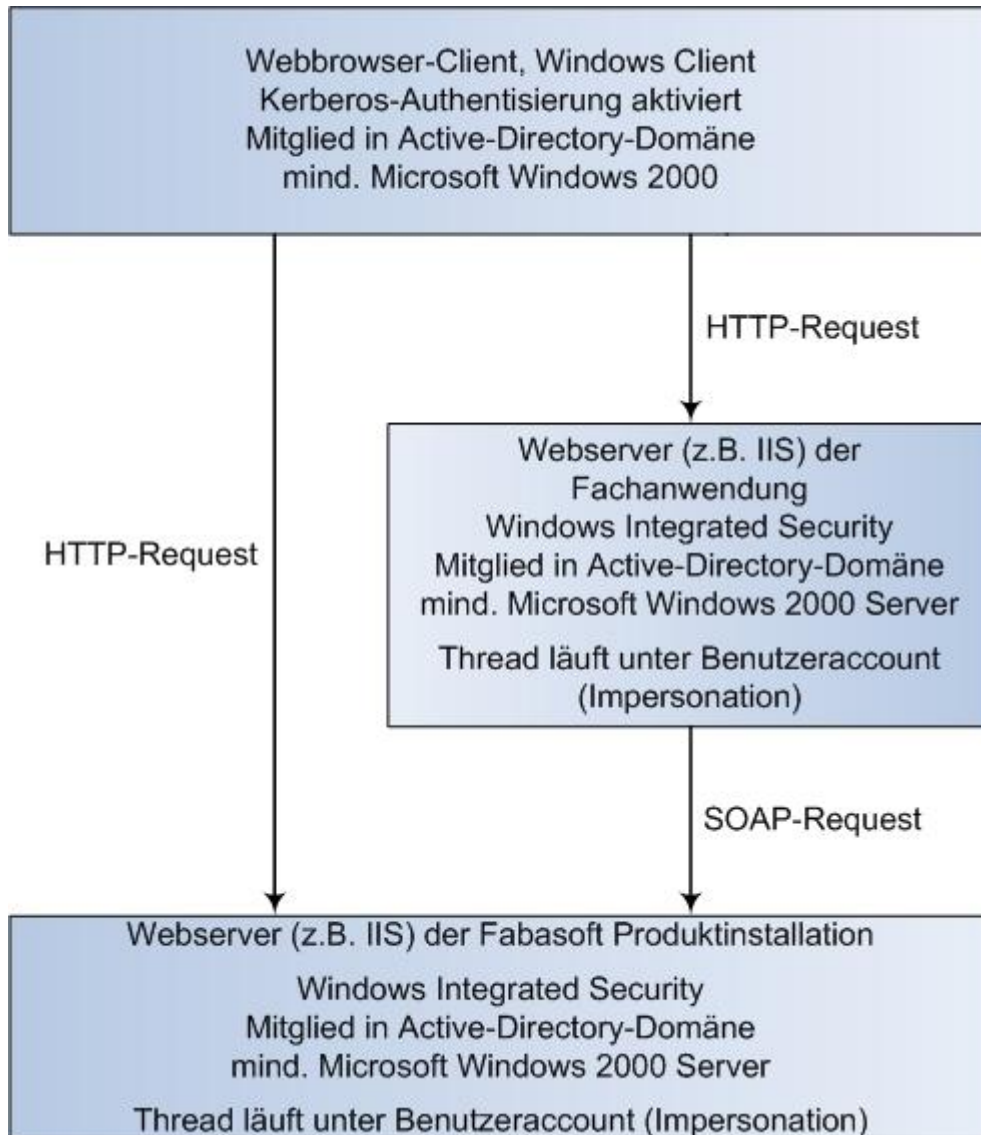
*Fachanwendungsdaten* können auch in der Stammdomäne bei Geschäftsobjekten gespeichert werden (siehe Kapitel 7.1 „Fachanwendungsdaten“.), wodurch eine Suche in diesen Daten in der Fabasoft Produktumgebung möglich ist. Die *Fachanwendungsdaten* werden als Inhalt in einem Fabasoft Folio MMC-Store gespeichert. Dadurch kann in diesen Daten durch die Volltextsuchfunktionalität der Fabasoft Folio MMC-Stores recherchiert werden.

## 9 Authentifizierung

Die Webservices können aus jenen Umgebungen heraus aufgerufen werden, die eine SOAP-Kommunikation (auf Basis HTTP/HTTPS und XML) erlauben. Diese Umgebung muss in der Lage sein, mit den Microsoft Internetinformationsdiensten via HTTP/HTTPS zu kommunizieren, insbesondere auf Basis der tatsächlich implementierten Security- und Authentifizierungskonfiguration der Microsoft Internetinformationsdienste.

Die Authentifizierung wird im Sinne eines Single Sign On technisch über den Mechanismus „Integrated Login“ realisiert.

Die Authentifizierung der Benutzer der Fachanwendung erfolgt beim „Integrated Login“ durch das Kerberosprotokoll, das sowohl die Fachanwendungsdomäne als auch die Stammdomäne unterstützen müssen. Der Zugriff erfolgt nach folgendem schematischem Ablauf:



#### Voraussetzungen für die Authentifizierung über Kerberos:

1. Fabasoft Produktinstallation
2. Entsprechende Einstellungen beim Webserver der Fachanwendung
3. Das Benutzerrecht für die Delegation muss entweder bei den Sicherheitseinstellungen des aktuellen Computers bzw. beim Benutzeraccount eingetragen sein (abhängig vom Account, unter dem der Webserver der Fachanwendung läuft).
4. Der Webbrowser muss Kerberos-Authentifizierung unterstützen.
5. Der betreffende angemeldete Anwender der Fachanwendung muss ein Benutzer in der Fabasoft Produktinstallation Domäne sein und auf dem Webserver der Fabasoft Produktumgebung authentisiert werden können.
6. Die Client-Verbindung zum Webservice der Fabasoft Produktumgebung muss mit Kerberos aufgebaut worden sein.

## 9.1 Active-Directory-Umgebung

Um die Kerberos-Authentifizierung zu verwenden, muss auf allen Rechnern eine unterstützte Microsoft-Windows-Version installiert sein. Darüber hinaus müssen die Benutzerkonten, die Rechte erhalten sollen, im Active Directory gespeichert werden.

Damit die Kerberos-Authentifizierung erfolgreich durchgeführt werden kann, müssen alle Rechner (Client- und Server-Maschinen) dem gleichen Active-Directory-Forest angehören.

## 9.2 Benutzerkonto für die Delegation konfigurieren

Folgende Schritte müssen am Active-Directory-Domänencontroller durchgeführt werden, damit sichergestellt wird, dass die Delegation für ein Benutzerkonto verwendet wird.

1. Loggen Sie sich als Administrator am Domänencontroller ein.
2. Klicken Sie auf „Start“ > „Programme“ > „Verwaltung“ > „Active Directory-Benutzer und – Computer“.
3. Navigieren Sie zum „Users“-Ordner der entsprechenden Microsoft Windows-Domäne.
4. Klicken Sie mit der rechten Maustaste auf das Benutzerkonto, das delegiert werden soll und wählen Sie „Eigenschaften“ aus dem Kontextmenü.
5. Wechseln Sie zur Registerkarte „Konto“.
6. Stellen Sie sicher, dass die Option „Konto ist vertraulich und kann nicht delegiert werden“ in der Liste „Kontooptionen“ nicht ausgewählt ist.
7. Klicken Sie auf „OK“, um das Dialogfenster zu schließen.

## 9.3 Benutzerkonto, unter dem die Fabasoft Folio Webservices laufen, konfigurieren

In diesem Abschnitt wird beschrieben, wie sichergestellt werden kann, dass für den Benutzer, unter dem die Fabasoft Folio Webservices (sowohl von der Fachanwendungsdomäne als auch von der Stammdomäne) laufen, die Delegation verwendet wird.

**Hinweis:** Bei Verwendung von Anwendungspools müssen die Einstellung für den Benutzer vorgenommen werden, unter dem der Anwendungspool läuft.

Durchzuführende Konfigurationseinstellungen am Active-Directory-Domänencontroller:

1. Klicken Sie auf „Start“ > „Programme“ > „Verwaltung“ > „Active Directory-Benutzer und – Computer“.
2. Navigieren sie zum „Users“-Ordner der entsprechenden Domäne.
3. Öffnen Sie die Eigenschaften des Benutzerkontos, unter dem die Fabasoft Folio Webservices bzw. der Anwendungspool läuft.
4. Aktivieren Sie die Option „Benutzer bei Delegation aller Dienste vertrauen (nur Kerberos)“ auf der Registerkarte „Delegation“.

Anschließend muss der Service Prinzipal Name für den Benutzer, unter dem die Fabasoft Folio Webservices laufen, gesetzt werden. Dadurch wird festgelegt, dass dieser Benutzer auf den angegebenen Rechnern als Service-Benutzer verwendet werden darf.

1. Geben Sie in der Kommandozeile folgenden Befehl ein, wobei `FQDNServerMachine` der vollständig qualifizierte Domänenname des Rechners, auf dem die Fabasoft Folio Webservices installiert sind, ist. `ServiceUser` ist der Domänen-Benutzeraccount.

```
setspn.exe -A HTTP/<FQDNServerMachine> <ServiceUser>
```

**Hinweis:** Es muss der vollständig qualifizierte Domänenname des Rechners verwendet werden, nicht der NetBIOS-Name!

## 10 SOAP-Aktionen der Fabasoft Produktinstallation

Die SOAP-Aktionen der Fabasoft Produktinstallation werden über die Fabasoft Folio Webservices verfügbar gemacht. Für die Nutzung der SOAP-Aktionen muss daher mindestens eine Fabasoft Folio Webservice Installation verfügbar sein.

**Hinweis:** Detaillierte Informationen zu unterstützten Betriebssystemen und unterstützter Software finden Sie im Readme-Dokument im Verzeichnis `Additions/Documents` auf Ihrer Fabasoft Produkt-CD.

SOAP ist ein XML-basiertes Protokoll für den Austausch von Informationen in einer dezentralisierten, verteilten Umgebung. SOAP-Nachrichten werden über die Schnittstelle der Fabasoft Produktinstallation mithilfe des Transportprotokolls HTTP übermittelt.

Die vollständige Beschreibung der einzelnen Webservices erfolgt über ein WSDL-Dokument (Web Service Description Language). Darin enthalten sind

1. Typdefinitionen der Parameter und Prototypen der aufrufbaren Methoden
2. Bindung an ein Transportprotokoll (HTTP)
3. Endpunkt des Service (Service-URL)

### 10.1 Technische Aspekte der SOAP-Schnittstelle

Die WSDL-Beschreibung eines Webservice der Fabasoft Produktinstallation wird dynamisch über einen parametrisierten URL erzeugt, auf den ein HTTP-GET-Request ausgeführt wird. Dieser URL beschreibt auch den Endpunkt des Webservice. Über einen HTTP-POST-Request werden SOAP-Nachrichten an das entsprechende Webservice übermittelt.

Die Verwendung der Webservices der Fabasoft Produktinstallation entspricht einem Request-Response-Modell. Um die Webservices anzusprechen, wird die „Document-Style“-Aufrufkonvention des SOAP-Protokolls verwendet. Das Webservice erhält über den SOAP-Request ein XML-Dokument als Input. Der SOAP-Response, der dem Aufruf folgt, enthält wiederum ein XML-Dokument, das die Output-Daten des Webservice enthält.

Falls bei der Ausführung des Webservice ein Fehler auftritt, wird im SOAP-Response ein SOAP-Fehler übermittelt, der die Fehlerbeschreibung des Webservice enthält.

Jeder Aufruf eines Webservice der Fabasoft Produktinstallation entspricht mindestens einer Transaktion, wobei jede Transaktion ein atomarer Schritt ist, der entweder zur Gänze ausgeführt wird oder nicht. Ein Aufruf eines Webservice kann auch mehrere Transaktionen umfassen; soweit nicht gegenteilig dokumentiert, ist ein Aufruf genau eine Transaktion.

Mehrere atomare SOAP-Aktionen können in einer Transaktion zusammengefasst werden. Eine Transaktion kann durch die letzte SOAP-Aktion abgeschlossen oder durch Aufruf der Aktion *Verwerfen einer SOAP-Transaktion* (`FSCOWS@1.1001:SOAPAbort`) abgebrochen werden.

### Beispiel:

In diesem Beispiel wird eine Transaktionsklammer verwendet. Innerhalb einer Transaktion werden ein Ordner und ein Word-Dokument erzeugt. Das Commit der Transaktion passiert hier vor dem Aufruf der letzten SOAP-Aktion.

```
FSC.SWC_1_1009_MyWebSvc svc = new FSC.SWC_1_1009_MyWebSvc();
svc.cootx = new FSC.TXContext();
...
res1 = svc.FSCBAI_1_1001_SOAPCreateObject(...); // Ordner erstellen
...

svc.cootx.commit = TRUE;
res2 = svc.FSCBAI_1_1001_SOAPCreateObject(...); //Word erstellen
...
```

## 10.2 Aufbau des Service-URL

Der URL, der zum Generieren des WSDL-Dokuments und als Service-Endpunkt dient, ist folgendermaßen aufgebaut:

`http://<webserver>/<vdir>/fscdav/wsdl?params`

`params` ist wie folgt zusammengesetzt:

```
params ::= param { "&" param }
param   ::= websvc | operation | mapping
websvc  ::= "websvc=" object
operation ::= "operation=" operationname
mapping  ::= "mapping=" object
operationname ::= <name of the operation to be executed>
object    ::= objectaddr | reference
objectaddr ::= <Fabasoft Folio object address>
reference  ::= <Fabasoft Folio reference>
```

Die Bedeutung der einzelnen Parameter ist in der folgenden Auflistung näher beschrieben. Parameter, die in eckiger Klammer ([]) angeführt werden, sind optional.

Parameter	Beschreibung
<code>websvc</code>	<i>Webservice-Definitionsobjekt</i> ( <code>FSCOWS@1.100:WebServiceDefinition</code> ), das die Aktionen des Service enthält. Dieser Parameter ist sowohl beim Generieren des WSDL-Dokuments als auch beim Ausführen einer SOAP-Aktion anzugeben. Nähere Informationen dazu siehe Kapitel 10.3 „Webservice-Definition“.

[operation]	Diese URL-Option ist nur beim Ausführen einer SOAP-Aktion relevant. Alternativ zum HTTP-Header „SOAPAction“ kann mit diesem URL-Parameter die auszuführende Operation angegeben werden.
[mapping]	Objektadresse oder <i>Referenz</i> einer XML-Abbildung. Falls angegeben, so wird das XML-Schema der Abbildung in das WSDL-Dokument generiert und die Abbildung selbst als Parameter an die SOAP-Aktion übergeben.

### 10.3 Webservice-Definition

Der `webservice`-Parameter eines Webservice-URL verweist auf ein Objekt der Klasse *Webservice-Definition* (`FSCOWS@1.1001:WebServiceDefinition`). In diesem Objekt sind eine Liste von SOAP-Aktionen und andere Einstellungen hinterlegt, die das Webservice bilden.

- **Kennung** (`FSCOWS@1.1001:webserviceidentifizier`)  
Alternative Bezeichnung des Webservices für SOAP-Clients. Falls nicht angegeben, so wird die Referenz der Webservice-Definition verwendet.
- **Verwende Transaktion** (`FSCOWS@1.1001:webserviceusetransaction`)  
Erlaubt es dem Client, mehrere SOAP-Aktionen in einer Transaktion zu bündeln.
- **Webservice-Aktionen** (`FSCOWS@1.1001:webserviceactions`)  
Aggregatsliste mit den aufrufbaren Aktionen. Ein Eintrag besteht aus folgenden Eigenschaften:
  - **Operation** (`FSCOWS@1.1001:webserviceoperation`)  
Alternative Bezeichnung der Aktion. Falls nicht angegeben, so wird die *Referenz* der Aktion (Eigenschaft `FSCOWS@1.1001:webserviceaction`) verwendet.
  - **Webservice-Aktion** (`FSCOWS@1.1001:webserviceaction`)  
Aktionen der Klassen *Aktion* (`COOSYSTEM@1.1:Action`) und *SOAP-Aktion* (`FSCOWS@1.1001:SOAPAction`) können angegeben werden.
  - **Objekt-Klasse** (`FSCOWS@1.1001:webserviceobjclass`)  
Diese Eigenschaft ist nur bei Aktionen der Klasse *SOAP-Aktion* (`FSCOWS@1.1001:SOAPAction`) relevant. Eine SOAP-Aktion kann mit einer Objektklasse qualifiziert werden, wodurch unterschiedliche XML-Zuordnungen für dieselbe SOAP-Aktion definiert werden können.
  - **XML-Zuordnung** (`FSCOWS@1.1001:webservicemapping`)  
In dieser Eigenschaft wird die anzuwendende XML-Zuordnung angegeben. XML-Zuordnungen werden nur für SOAP-Aktionen (Objektklasse `FSCOWS@1.1001:SOAPAction`) berücksichtigt. Mithilfe von XML-Zuordnungen werden die benötigten Parameter aus dem eingehenden XML-Datenstrom extrahiert bzw. es wird der ausgehende XML-Datenstrom generiert. Siehe auch zuvor beschriebene Eigenschaft *Objekt-Klasse*.

XML-Zuordnungen können an mehreren Stellen definiert werden. Die Priorisierung erfolgt gemäß folgender Liste:

1. Höchste Priorität hat der `mapping`-Parameter des Webservice-URL. So besteht die Möglichkeit, beim Aufruf einer SOAP-Aktion die XML-Zuordnung vorzugeben.
2. XML-Zuordnung in der *Webservice-Definition*.

3. XML-Zuordnung, die durch einen Pre-Wrapper der Aktion *Ermittle das tatsächliche Mapping einer SOAP-Aktion* (`FSCOWS@1.1001:GetMapping`) geliefert wird.
4. Die XML-Zuordnung der SOAP-Aktion (Eigenschaft *XML-Zuordnung* (`COOXML@1.1:xmlactmapping`)) hat die niedrigste Priorität.

Es besteht auch die Möglichkeit Aktionen (Objektklasse `COOSYSTEM@1.1:Action`) in die Webservice-Definition einzuhängen. In diesem Fall erfolgt eine automatische Zuordnung der Aktionsparameter zu SOAP-Nachrichten.

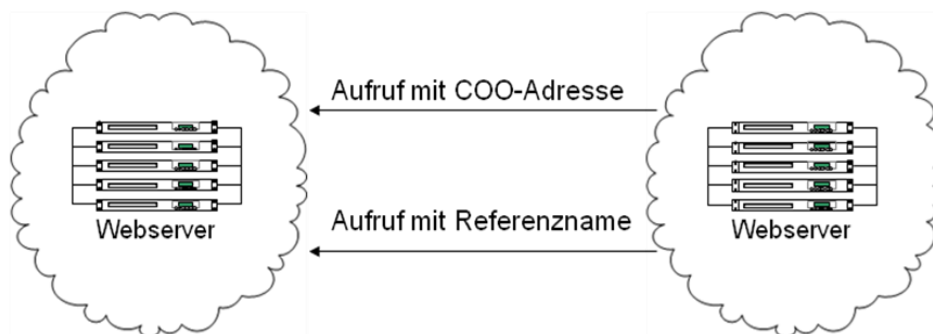
Weiters enthält ein Webservice-Definitionsobjekt Informationen für die Registrierung des Webservices in einem UDDI-Verzeichnis (*Universal Description Discovery and Integration*). Die Registrierung bzw. Deregistrierung erfolgt über die Menübefehle „Registrierte in UDDI-Datenbank“ bzw. „Entferne Registrierung von UDDI-Datenbank“.

### Beispiel:

Folgende Beispiele beschreiben verschiedene Möglichkeiten, ein konkretes Webservice (hier das Service `SWC@1.1001:WebService` mit der Objektadresse `COO.1.1001.1.2049569`) aufzurufen:

#### Fabasoft Produktinstallation

#### Fachanwendung



```
http://<webserver>/<vdir>/fscdav/wsdl?websvc=COO.1.1001.1.2049569
&style=doc
```

```
http://<webserver>/<vdir>/fscdav/wsdl?websvc=SWC@1.1001:WebService
&style=doc
```

Im ersten Beispiel wird das Webservice über die Objektadresse (`COO.1.1001.1.2049569`) angesprochen. Das zweite Beispiel benutzt die Referenz des Webservices (`SWC@1.1001:WebService`).

Ab der Version Fabasoft Folio 2017 wird die Beschreibung der Parameter (`actparoptional`) ausgewertet, bis dahin waren alle Parameter optional.

# 11 Funktionsteilbereich „Aufruf von Funktionalität der Fabasoft Produktinstallation durch eine Fachanwendung“

## 11.1 Lösungsbereich Transaktionen

### 11.1.1 Mehrere SOAP-Aktionen zu einer Transaktion zusammenfassen

Die Fabasoft Produktinstallation bietet einer Fachanwendung die Möglichkeit, mehrere von der Fachanwendung aufgerufene SOAP-Aktionen in einer atomaren Transaktion in der Fabasoft Produktinstallation abzuarbeiten.

Zum Beispiel könnten folgende SOAP-Aktionen in einer Transaktion durchgeführt werden:

1. Erzeugen eines Ordners
2. Erzeugen eines Word-Objekts

Der jeweilige Transaktionskontext der Installation der Fabasoft Produktinstallation wird im SOAP-Header zwischen der Fabasoft Produktinstallation und der Fachanwendung transportiert.

## 11.2 Lösungsbereich Fachanwendungsdaten

Im Folgenden wird der Zugriff auf die *Fachanwendungsdaten* im Aggregat *Fachanwendungsdaten* (FSCBAI@1.1001:data) mittels SOAP-Aktionen beschrieben. Diese SOAP-Aktionen werden von der Softwarekomponente *Integration for Business Application* (FSCBAI@1.1001) zur Verfügung gestellt.

### 11.2.1 XML-Fachanwendungsdaten schreiben

Die SOAP-Aktion *Setzen der Fachanwendungsdaten über XML/SOAP* (FSCBAI@1.1001:SOAPSetData) der Fabasoft Produktinstallation bietet einer Fachanwendung die Möglichkeit die XML-Fachanwendungsdaten des betreffenden Objekts zu schreiben.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:bai:businessappdata

```
<SOAPSetDataRequest>
  <objaddress> Objektadresse </objaddress>
  <dept> Referenz der Fachanwendungsbereichsdefinition </dept>
  <data> Fachanwendungsdaten </data>
  <xpath> XPath-Ausdruck </xpath>
</SOAPSetDataRequest>
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:bai:businessappdata

---

```
<SOAPSetDataResponse>
  <resultstring> Ergebnis der SOAP-Aktion </resultstring>
</SOAPSetDataResponse>
```

---

Wird im XPath-Ausdruck nichts angegeben, so wird der gesamte XML-Inhalt aus dem Fachanwendungsaggregat überschrieben. Mithilfe des XPath-Ausdrucks kann definiert werden, ob nur ein Teilbaum der XML-Struktur überschrieben wird. Die Daten, die in den definierten Knoten geschrieben werden sollen, werden im Knoten `<data>` angegeben.

In dem XPath-Ausdruck können Namespaces verwendet werden, die beim übergebenen *Fachanwendungsbereich* definiert sein müssen.

### Beispiel:

Von einem Objekt soll ein XML-Knoten der Fachanwendungsdaten überschrieben werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Fachanwendungsdaten setzen* (FSCBAI@1.1001:SOAPSetData) übermittelt.

```
::params=coort.CreateDictionary();
::params.SetEntryValue("objaddress",0,this.objaddress);
::params.SetEntryValue("dept",0,::deptref);
::data=coort.CreateDictionary();
::filesobj=coort.CreateDictionary();
::filesobj.SetEntryValue("Testwert",0,"Value");
::params.SetEntryValue("data",0,::filesobj);
::params.SetEntryValue("xpath",0,"//ns1:Incoming/ns1:filesobj");
::xmlcontent=coort.CreateContent(this);
FSCBAI@1.1001:CallSoapXmlEx(::dept,"SoapSetData",::params,&::xmlcontent)
```

## 11.2.2 XML-Fachanwendungsdaten lesen

Die SOAP-Aktion *Ermitteln der Fachanwendungsdaten über XML/SOAP* (FSCBAI@1.1001:SOAPGetData) der Fabasoft Produktinstallation bietet einer Fachanwendung die Möglichkeit die XML-Fachanwendungsdaten des betreffenden Objekts zu lesen.

---

### Input

---

**Namespace:** urn:schemas-fabasoft-com:bai:businessappdata  
<SOAPGetDataRequest>  
 <objaddress> Objektadresse </objaddress>  
 <dept> Referenz der Fachanwendungsbereichsdefinition </dept>  
 <xpath> XPath-Ausdruck </xpath>  
</SOAPGetDataRequest>

---

### Output

---

**Namespace:** urn:schemas-fabasoft-com:bai:businessappdata  
<SOAPGetDataResponse>  
 <resultstring> Ergebnis der SOAP-Aktion </resultstring>

---

```
<data> Fachanwendungsdaten </data>
</SOAPGetDataResponse>
```

Wird im XPath-Ausdruck nichts angegeben, so wird der gesamte XML-Inhalt aus dem Fachanwendungsaggregat als XML-Struktur ausgegeben. Mithilfe des XPath-Ausdrucks kann festgelegt werden, dass nur ein Teilbaum der XML-Struktur ausgegeben wird. Im XPath-Ausdruck können XML-Namespaces verwendet werden, die beim übergebenen *Fachanwendungsbereich* definiert sein müssen.

Der Response enthält im XML-Knoten `<data>` den ermittelten Knoten mit dem Wert bzw. den ermittelten Teilbaum mit alle untergeordneten Knoten.

#### **Beispiel:**

Von einem Objekt soll ein XML-Knoten der Fachanwendungsdaten ausgegeben werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Fachanwendungsdaten lesen* (FSCBAI@1.1001:SOAPGetData) übergeben.

```
::params=coort.CreateDictionary();
::params.SetEntryValue("objaddress",0,this.objaddress);
::params.SetEntryValue("dept",0,::deptref);
::params.SetEntryValue("xpath",0,"//ns1:Incoming/ns1:filesobj");
::xmlcontent=coort.CreateContent(this);
FSCBAI@1.1001:CallSoapXmlEx(::dept,"SoapGetData",::params,&::xmlcontent)
```

### **11.2.3 Fachdokument in der Fabasoft Produktinstallation anzeigen**

Fachdokumente sind Dokumente im HTML-Format, die mittels einer XSL-Transformation aus den XML-Fachanwendungsdaten erstellt und für den Anwender in der Fabasoft Produktinstallation angezeigt werden können. Pro Fachanwendungsbereich und Objektklasse wird ein definiertes XSLT-Stylesheet verwendet.

Der Aufruf dieser Funktionalität durch den Anwender in der Installation der Fabasoft Produktinstallation erfolgt durch einen entsprechend konfigurierten Kontextmenübefehl des Objekts.

### **11.2.4 Fachanwendungsdaten in Dokument-Eigenschaften verwenden**

Blatt-Knoten des XML-Inhalts von Fachanwendungsdaten können bei geeigneter Definition als Text in Dokument-Eigenschaften von Microsoft Office-Dokumenten verwendet werden.

## **11.3 Lösungsbereich Objekt-Bearbeitung**

Die Bearbeitung von DMS(Dokument Management System)-Objekten umfasst das Erzeugen und Bearbeiten von Objekten mit Multimedia-Inhalt. Dieser Inhalt kann mit einem von der Fabasoft Produktinstallation unterstützten Produktivitätswerkzeug (z. B. Microsoft Office Word) gelesen und bearbeitet werden. Weiters werden SOAP-Aktionen zur Verfügung gestellt, die Versionen erstellen und bereits erstellte Version von Fabasoft Folio Objekten behandeln können.

### 11.3.1 Objekt bearbeiten

Eine Fachanwendung kann unter Verwendung der Fabasoft vApp-Technologie URLs eines Webservice der Fabasoft Produktinstallation aufrufen, um die entsprechende Anwendung für ein angegebenes Objekt aufzurufen.

Ein Aufruf an die Fabasoft Produktinstallation aus der Fachanwendung muss nach folgendem Schema aufgebaut sein:

Parametername	IN/OUT	Typ	Erklärung
ax	In	STRING	<p>Definiert eine Anwendung.</p> <p>Beispielanwendung für Aufruf an die Fabasoft Produktinstallation: FSCVAPP@1.1001:StartMenu (Objektadresse COO.1.1001.1.91492)</p> <p>Bei einem Aufruf an eine Fachanwendung kann dieser Parameter in der <i>Konfiguration für Fachanwendungen</i> in der Eigenschaft <i>Fachanwendungsaufrufe</i> im <i>Ausdruck für GUI-Aktion (URL)</i> angegeben werden.</p>
sys_action	In	STRING	<p>Definiert eine Aktion.</p> <p>Beispielaktion für Aufruf an die Fabasoft Produktinstallation: <i>Eigenschaften der markierten Objekte bearbeiten</i> (Objektadresse COO.1.1.1.1066)</p> <p>Bei einem Aufruf an eine Fachanwendung kann dieser Parameter in der <i>Konfiguration für Fachanwendungen</i> in der Eigenschaft <i>Fachanwendungsaufrufe</i> im <i>Ausdruck für optionale GUI-Parameter</i> angegeben werden.</p>
commit	In	STRING	TRUE
sys_object	In	STRING	<p>Die eindeutige Kennung des Geschäfts- oder Fachobjekts.</p> <p>Bei einem Aufruf an eine Fachanwendung wird dieser Parameter automatisch aufgrund des beim jeweiligen Objekt gespeicherten Fremdschlüssels und des Fachanwendungsbereichs gesetzt.</p>
ru	In	STRING	<p>Der Return-URL, der nach dem Ende der Bearbeitung in der Fachanwendung aufgerufen wird.</p> <p>Bei einem Aufruf an eine Fachanwendung kann dieser Parameter in der <i>Konfiguration</i></p>

für *Fachanwendungen* in der Eigenschaft *Fachanwendungsaufrufe* im Ausdruck für *Return-URL (Post-GUI)* als Expression angegeben werden.

<name>	In	STRING	Sonstige URL-Parameter als <expression>. Bei einem Aufruf an eine Fachanwendung kann dieser Parameter in der <i>Konfiguration für Fachanwendungen</i> in der Eigenschaft <i>Fachanwendungsaufrufe</i> im Ausdruck für <i>optionale GUI-Parameter</i> angegeben werden.
--------	----	--------	---

### Beispiel:

Aufruf an eine Fabasoft Fachanwendung:

```
http://<webserver>/<vdir>/fscasp/content/bin/fscvext.dll?ax=COO.1.1001.1.91492&sys_action=COO.1.1.1.1066&commit=TRUE&ru=javascript:window.close()&sys_object=COO.1.1094.2.1002276
```

## 11.3.2 DMS-Dokument erzeugen

### Durch die Fabasoft Produktinstallation

In der Fabasoft Produktinstallation wird ein Schriftstück erzeugt.

### Durch die Fachanwendung

Wird ein Dokument in der Fachanwendung erstellt, wird dazu die fachspezifische Funktionalität genutzt. Anschließend kann das dabei erstellte Dokument mittels SOAP in die Stammdomäne geladen werden.

### Erzeugen von Inhaltsobjekten via SOAP (FSCOWS@1.1001:SOAPCreateContentObjects)

#### Input

Namespace: urn:schemas-fabasoft-com:ows:createcontentobjects

```
<createcontentobjects>
  <objectclass> Objektklasse des Dokuments </objectclass>
  <contentlist> Base64-kodierte(r) Inhalt(e)
  ...
</contentlist>
  <sourceextension> Erweiterung Originalinhalt </sourceextension>
</createcontentobjects>
```

#### Output

Namespace: urn:schemas-fabasoft-com:ows:createcontentobjects

```
<createcontentobjectsresult>
  <objectlist> Objektadressen der erzeugten Objekte
```

```
...
</objectlist>
</createcontentobjectsresult>
```

---

### 11.3.3 Inhalt eines DMS-Objekts bearbeiten

Eine Fachanwendung kann unter Verwendung des standardisierten Protokolls WebDAV Inhalte („Contents“), die in der Fabasoft Produktinstallation verwaltet werden, adressieren. Analog zum Use-Case „Objekt bearbeiten“ ruft die Fachanwendung in diesem Fall einen URL eines Webservice der Fabasoft Produktinstallation auf.

WebDAV ("Web based Distributed Authoring and Versioning") ist eine standardisierte Erweiterung des HTTP-Protokolls. Zusätzlich zu den Standard-HTTP-Funktionen für Download, Upload usw. sind Funktionen zum Sperren/Entsperren von Ressourcen (LOCK/UNLOCK) und Abfragen/Setzen/Suchen von Eigenschaften (PROPFIND/PROPPATCH/SEARCH) definiert.

Die allgemeine Syntax eines WebDAV-URL lautet:

```
http://<webserver>/<vdir>/FSCDAV/<FlatSpace>
```

```
FlatSpace          ::= Prefix "?" [ Parameters ] [ "/" Postfix ] .
Parameters         ::= Parameter { "&" Parameter } .
Prefix            ::= "READONLY" | "DAV" .
Postfix           ::= <any file name with extension>
Parameter         ::= ObjParam | ContentParam | ActionParam | DownloadParam |
                    ux | px | gx | xx | dx | expires | ArgParam | TypParam |
                    ResParam
ObjParam          ::= "OBJ" "=" CooAddr .
ContentParam      ::= "CONTENT" "=" AttrPath .
DownloadParam     ::= "DOWNLOAD" "=" FileName .
ActionParam       ::= "ACTION" "=" CooAddr .
ArgParam          ::= "ARG" [ number ] "=" QuotedStr .
TypParam          ::= "TYP" [ number ] "=" CooAddr .
ResParam          ::= "RES" "=" "ARG" [ number ] .
expires           ::= "EXPIRES" "=" seconds .
CooAddr           ::= <object address> .
AttrPath          ::= AttrRef "(" Index ")" { AttrRef "(" Index ")" } .
AttrRef           ::= <reference of the attribute> .
FileName          ::= <file name with extension> .
```

Anwendbare HTTP/WebDAV-Methoden:

---

Command	HTTP/WebDAV-Methoden	Bedeutung
---------	----------------------	-----------

---

READONLY	GET	Lesen von Inhalten
DAV	LOCK, UNLOCK, GET, PUT	Bearbeiten von Inhalten

Anmerkungen:

- Die Kommandos READONLY und DAV verfügen über die Parameter OBJ und CONTENT, wobei CONTENT optional ist. Der Standardwert dafür ist "content(0)contcontent(0)". In den meisten Fällen kann dieser Standardwert verwendet werden, da damit der Hauptinhalt eines Objekts mit Inhalt beschrieben wird.
- Das Kommando READONLY kann mit dem optionalen Parameter DOWNLOAD ausgeführt werden. Dieser Parameter bewirkt, dass ein Download-Dialog geöffnet wird. Der vorgeschlagene Dateiname ist der Wert des DOWNLOAD-Parameters (**Beispiel:** DOWNLOAD=test.doc).  
**Achtung:** Im Microsoft Windows-Explorer kann pro Dateityp angegeben werden, ob für ein heruntergeladenes Dokument ein "Save/Open"-Dialog angezeigt wird, oder ob das Dokument sofort geöffnet werden soll ("Confirm open after download"). Weiters kann angegeben werden, ob ein Dokument in-process oder out-of-process geöffnet werden soll ("Browse in same window").
- Mit dem optionalen Postfix kann angegeben werden, wie der Webbrowser die ankommenden Daten interpretiert. Beispielsweise bewirkt "x.doc" als Postfix, dass der Webbrowser automatisch Microsoft Office Word öffnet. Wird kein Postfix angegeben, so wird versucht, aus dem Content-Aggregat bzw. über die Funktion `CCooContent::GetFile()` eine passende Dateiendung zu bestimmen. Aus der Dateiendung wird ein MIME-Type berechnet und als Content-Type-Header an den Webbrowser zurückgegeben.
- Der optionale Parameter EXPIRES legt fest, welches Ablaufdatum der gelieferte Inhalt haben soll. Der Parameter-Wert gibt die Anzahl der Sekunden ab dem aktuellen Zeitpunkt an. Wird dieser Parameter nicht angegeben, so gilt der gelieferte Inhalt als bereits abgelaufen (HTTP-Header "Expires: 0"). Wenn der Client diesen URL erneut referenziert, so wird der Inhalt erneut vom Server angefordert.
- Das DAV-Kommando sollte nur dann verwendet werden, wenn der Inhalt von einer Applikation bearbeitet wird. In allen anderen Fällen ist das READONLY-Kommando vorzuziehen.
- Die Reihenfolge der Parameter (OBJ, CONTENT, ARG, DOWNLOAD usw.) ist irrelevant.

### Beispiel:

WebDAV-Aufrufe:

```

http://<webserver>/<vdir>/FSCDAV/DAV?OBJ=COO.1.137.2.222&CONTENT=content(0)contcontent(0)
http://<webserver>/<vdir>/FSCDAV/DAV?OBJ=COO.1.137.2.222&CONTENT=content(0)contcontent(0)&px=COO.1.1001.1.63575&gx=COO.1.1001.1.63762
http://<webserver>/<vdir>/FSCDAV/READONLY?OBJ=COO.1.137.2.222&CONTENT=content(0)contcontent(0)
http://<webserver>/<vdir>/FSCDAV/READONLY?OBJ=COO.1.137.2.222&CONTENT=content(0)contcontent(0)/x.doc
http://<webserver>/<vdir>/FSCDAV/READONLY?OBJ=COO.1.137.2.223/x.xls

```

http://<webserver>/<vdir>/FSCDAV/READONLY?OBJ=COO.1.137.2.225&CONTENT=COODESK\_1\_1\_o  
bjthumbnail(0)/x.bmp  
http://<webserver>/<vdir>/FSCDAV/READONLY?OBJ=COO.1.137.2.226&CONTENT=COODI\_1\_1001\_  
imgdoccontents(0)COODI\_1\_1001\_imgcont(0)/x.tif  
http://<webserver>/<vdir>/FSCDAV/READONLY?OBJ=COO.1.137.2.222&CONTENT=content(0)con  
tcontent(0)&DOWNLOAD=fsc.doc

### 11.3.4 Versionierung

Die Fabasoft Produktinstallation stellt vier SOAP-Aktionen zum Lesen und Bearbeiten von Versionen über SOAP zur Verfügung:

- *Version über XML/SOAP erzeugen* (FSCBAI@1.1001:SOAPCreateVersion)
- *Version über XML/SOAP lesen* (FSCBAI@1.1001:SOAPReadVersion)
- *Version über XML/SOAP wiederherstellen* (FSCBAI@1.1001:SOAPRestoreVersion)
- *Version über XML/SOAP vernichten* (FSCBAI@1.1001:SOAPDeleteVersion)

#### Version über XML/SOAP erzeugen (FSCBAI@1.1001:SOAPCreateVersion)

Diese SOAP-Aktion erstellt eine neue Version des angegebenen Objekts.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:bai:version

```
<SOAPCreateVersionRequest>  
  <objid> Objektadresse </objid>  
  <autofixed> Wurde die Version automatisch erstellt </autofixed>  
  <description> Beschreibung </description>  
  <convertfinalform> Konvertierung ins finale Format </convertfinalform>  
  <finalform> Finales Format </finalform>  
  <noautopurge> Automatisches Löschen </noautopurge>  
</SOAPCreateVersionRequest>
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:bai:version

```
<SOAPCreateVersionResponse>  
  <version>  
    <newversnr> Laufende Nummer der erstellten Version </newversnr>  
    <createdat> Erstellungsdatum der Version </createdat>  
    <savedby> Objektadresse des Userobjekts des Erstellers </savedby>  
  </version>  
</SOAPCreateVersionResponse>
```

Diese SOAP-Aktion erstellt eine neue Version und legt sie im entsprechenden Fabasoft Folio MMC-Store ab. Das Request-XML kann mit mehreren Parametern befüllt werden, wobei nur die XML-Knoten <objid> des betreffenden Objekts und <description> benötigt werden, um eine Version erfolgreich erzeugen zu können.

Der XML-Knoten `<autofixed>` dient zur Information und beschreibt, ob die Version automatisch erstellt wurde oder nicht.

Weiters kann eine Version in ein finales Format konvertiert werden. Dies wird mit den beiden XML- Knoten `<convertfinalform>` und `<finalform>` ermöglicht. Ist `<convertfinalform>` `TRUE`, wird das Zielformat aus `<finalform>` ausgelesen. Wird ein Zielformat gefunden, wird versucht, eine Konvertierung anzustoßen. Wird kein Zielformat übergeben, wird es aus der aktuellen Domäne ermittelt.

### Beispiel:

In der Fabasoft Produktinstallation soll ein Objekt versioniert und in PDF konvertiert werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (`FSCBAI@1.1001:CallSoapXmlEx`) an die SOAP-Aktion *Version über XML/SOAP erzeugen* (`FSCBAI@1.1001:SOAPCreateVersion`) übermittelt.

```
::params=coort.CreateDictionary();
::returnval=coort.CreateDictionary();

::params.SetEntryValue("objid",0,this.objaddress);
::params.SetEntryValue("description",0,"Das ist ein SOAP-Test");
::params.SetEntryValue("autofixed",0,TRUE);
::params.SetEntryValue("convertfinalform",0,TRUE);
::params.SetEntryValue("finalform",0,"pdf");
::params.SetEntryValue("noautopurge",0,TRUE);
FSCBAI@1.1001:CallSoapXmlEx(::dept,"CreateVersion",::params,&::returnval);
```

Die Adresse des zu versionierenden Objekts wird im XML-Knoten `<objid>` hinterlegt. In diesem Fall wurde die Version automatisch vom System erstellt und darf auch ohne Nachfrage gelöscht werden. Weiters wurde eine Konvertierung in PDF angestoßen.

### Version über XML/SOAP lesen (`FSCBAI@1.1001:SOAPReadVersion`)

Mit dieser SOAP-Aktion kann eine Version eines Fabasoft Folio Objekts gelesen werden.

---

#### Input

**Namespace:** `urn:schemas-fabasoft-com:bai:version`

```
<SOAPReadVersionRequest>
  <objid> Objektadresse </objid>
  <dept> Fachanwendungsbereich </dept>
  <date> Datum der Version </date>
</SOAPReadVersionRequest>
```

---

#### Output

**Namespace:** `urn:schemas-fabasoft-com:bai:version`

```
<SOAPReadVersionResponse>
  <resultstring> Ergebnis der SOAP-Aktion </resultstring>
```

```
<data> Daten des Objekts </data>
</SOAPReadVersionResponse>
```

---

Diese SOAP-Aktion kann verwendet werden, um eine Version eines Objekts zu lesen. Im Request-XML der SOAP-Aktion wird im XML-Knoten `<objid>` das Objekt angegeben, von dem die Version gelesen werden soll. Der Fachanwendungsbereich muss definiert werden, da über diesen das XML-Mapping in der Konfiguration für Fachanwendungen gesucht wird. Mit dem XML-Mapping wird der XML-Knoten `<data>` des Response-XML aufbereitet. Im XML-Knoten `<date>` wird das Datum der zu lesenden Version angegeben.

### Beispiel:

In der Fabasoft Produktinstallation soll eine Version eines Objekts gelesen werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Version über XML/SOAP erzeugen* (FSCBAI@1.1001:SOAPReadVersion) übermittelt.

```
::params=coort.CreateDictionary();
::returnval=coort.CreateContent();

::params.SetEntryValue("objid",0,this.objaddress);
::params.SetEntryValue("dept",0,::dept);
::params.SetEntryValue("date",0,"2005-04-21T14:00:00");
FSCBAI@1.1001:CallSoapXmlEx(::dept,"ReadVersion",::params,&::returnval);
```

Die Adresse des Objekts, von dem die Version gelesen werden soll, wird im XML-Knoten `<objid>` hinterlegt. Über den Fachanwendungsbereich wird das entsprechende XML-Mapping ermittelt, das zur Aufbereitung des Knotens `<data>` des XML-Response verwendet wird.

### Version über XML/SOAP wiederherstellen (FSCBAI@1.1001:SOAPRestoreVersion)

Mit dieser SOAP-Aktion kann eine Version wiederhergestellt werden. Die aktuelle Version des Objekts wird mit der im SOAP-Aufruf definierten Version ersetzt.

---

## Input

---

**Namespace:** urn:schemas-fabasoft-com:bai:version

```
<SOAPRestoreVersionRequest>
  <objid> Objektadresse </objid>
  <date> Datum der Version </date>
</SOAPRestoreVersionRequest>
```

---

## Output

---

**Namespace:** urn:schemas-fabasoft-com:bai:version

```
<SOAPRestoreVersionResponse>
  <resultstring> Ergebnis der SOAP-Aktion </resultstring>
</SOAPRestoreVersionResponse>
```

---

Dieser SOAP-Aktion muss die Objektadresse des entsprechenden Objekts im XML-Knoten `<objid>` und das Datum der Version im XML-Knoten `<date>` übergeben werden. Der Inhalt des Objekts wird in der Fabasoft Produktinstallation mit der angegebenen Version ersetzt.

#### Beispiel:

In der Fabasoft Produktinstallation soll eine Version eines Objekts wiederhergestellt werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Version über XML/SOAP erzeugen* (FSCBAI@1.1001:SOAPRestoreVersion) übermittelt.

```
::params=coort.CreateDictionary();  
::returnval=coort.CreateContent();  
  
::params.SetEntryValue("objid",0,this.objaddress);  
::params.SetEntryValue("date",0,"2005-04-21T14:00:00");  
  
FSCBAI@1.1001:CallSoapXmlEx(,:"dept","RestoreVersion",,::params,&::returnval)
```

Die Adresse des Objekts wird im XML-Knoten `<objid>` hinterlegt. Das Datum definiert die Version, die wiederhergestellt wird.

#### Version über XML/SOAP vernichten (FSCBAI@1.1001:SOAPDeleteVersion)

Diese SOAP-Aktion vernichtet eine definierte Version eines Objekts.

---

### Input

**Namespace:** urn:schemas-fabasoft-com:bai:version

```
<SOAPDeleteVersionRequest>  
  <objid> Objektadresse </objid>  
  <date> Datum der Version </date>  
</SOAPDeleteVersionRequest>
```

---

### Output

**Namespace:** urn:schemas-fabasoft-com:bai:version

```
<SOAPDeleteVersionResponse>  
  <resultstring> Ergebnis der SOAP-Aktion </resultstring>  
</SOAPDeleteVersionResponse>
```

Dieser SOAP-Aktion muss die Objektadresse des entsprechenden Objekts im XML-Knoten `<objid>` und das Datum der Version im XML-Knoten `<date>` übergeben werden. Die definierte Version wird vernichtet.

#### Beispiel:

In der Fabasoft Produktinstallation soll eine Version eines Objekts vernichtet werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über*

*Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Version über XML/SOAP erzeugen* (FSCBAI@1.1001:SOAPDeleteVersion) übermittelt.

```
::params=coort.CreateDictionary();
::returnval=coort.CreateContent();

::params.SetEntryValue("objid",0,this.objaddress);
::params.SetEntryValue("date",0,"2005-04-21T14:00:00");

FSCBAI@1.1001:CallSoapXmlEx(::dept,"DeleteVersion",::params,&::returnval)
```

Die Adresse des Objekts wird im XML-Knoten <objid> hinterlegt. Das Datum definiert die Version, die vernichtet wird.

### 11.3.5 Neues Objekt erzeugen

Die SOAP-Aktion *Objekt über XML/SOAP erzeugen und Eigenschaften setzen* (FSCBAI@1.1001:SOAPCreateObject) bietet die Möglichkeit, ein neues Objekt zu erzeugen.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:bai:object

```
<SOAPCreateObjectRequest>
  <objclass> Referenz der Objektklasse </objclass>
  <objname> Name des Objekts </objname>
  <containerlist>
    <container> //n-mal
      <contaddress> Adresse des Containers </contaddress>
      <contattr> Zieleigenschaft des Containers </contattr>
    </container>
  </containerlist>
  <initialization>
    <mapping> Objektadresse des XML-Mapping </mapping>
    <dept> Referenz des Fachanwendungsbereichs </dept>
    <data>
      XML-Teilbaum
    </data>
  </initialization>
</SOAPCreateObjectRequest >
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:bai:object

```
<SOAPCreateObjectResponse>
  <objaddress> Objektadresse des erzeugten Objekts
</objaddress>
  <objname> Objektname des erzeugten Objekts </objname>
  <mappedcontent> XML, das durch das angegebene XML-Mapping erstellt wurde
</mappedcontent>
</SOAPCreateObjectResponse>
```

Die Aktion erzeugt ein neues Objekt der Objektklasse, die im XML-Element <objclass> angegeben ist. Falls zusätzlich zu den Informationen zum Erzeugen des Objekts

Initialisierungsdaten im Knoten `<initialization>/<data>` angegeben sind (statt dem `<any>`-Block), so werden diese mittels einer XML-Abbildung in das neu erzeugte Objekt geladen. Zur Ermittlung des Mappings gibt es zwei Möglichkeiten:

1. Die XML-Abbildung wird aus der Aggregatsliste *Schematazuordnung* aus der *Konfiguration für Fachanwendungen* ausgelesen. Dafür muss im Request-XML `<initialization>/<dept>` mit der Referenz eines Fachanwendungsbereichs befüllt werden. Anschließend sieht die Aktion in der Konfiguration nach und sucht nach einem Eintrag mit folgenden Kriterien:
  - **Objektklasse:** Objektklasse aus der XML-Node `<objclass>`
  - **Auslöser:** `FSCBAI@1.1001:SOAPCreateObject`
  - **Fachanwendungsbereich:** Fachanwendungsbereich aus der XML-Node `<initialization>/<dept>`
  - **Zuordnung:** Eingetragenes XML-Mapping
2. Die XML-Abbildung wird direkt in der XML-Node `<initialization>/<mapping>` der SOAP-Aktion übergeben.

Die Aktion liefert als Ergebnis ein XML-Dokument, das die Objektadresse, den Objektnamen und das Erstellungsdatum enthält.

#### Beispiel:

Anhand eines in der *Konfiguration für Fachanwendungen* eingetragenen XML-Mappings soll ein neues Objekt erstellt und mit Metadaten initialisiert werden. Weiters soll das Objekt in einem Ordner abgelegt werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (`FSCBAI@1.1001:CallSoapXmlEx`) an die SOAP-Aktion *Objekt über XML/SOAP erzeugen und Eigenschaften setzen* (`FSCBAI@1.1001:SOAPCreateObject`) übermittelt.

```
::params=coort.CreateDictionary();
::init=coort.CreateDictionary();
::containerlist=coort.CreateDictionary();
::container=coort.CreateDictionary();

::params.SetEntryValue("objclass",0,"COOMSOFFICE@1.1:WinWordObject");
::params.SetEntryValue("objname",0,"Erzeugtes Wordobjekt");

::container.SetEntryValue("contaddress",0,"COO.1.1065.1.165");
::container.SetEntryValue("contattr",0,"COOSYSTEM@1.1:objchildren");
::containerlist.SetEntryValue("container",0,:: container);

::init.SetEntryValue("dept",0,"FSCBANNEW@10.531:DeptA ");
::init.SetEntryValue("data",0,"XML-Teilbaum");

::params.SetEntryValue("containerlist",0,::containerlist);
::params.SetEntryValue("initialization",0,::init);

FSCBAI@1.1001:CallSoapXmlEx(::dept,"CreateObject",::params,&::returnval);
```

## 11.4 Lösungsbereich Workflow

Die Fabasoft Produktinstallation stellt mit der Softwarekomponente *Integration for Business Application: Workflow* (FSCBAIWF@1.1001) SOAP-Aktionen zum Steuern und Erstellen von Prozessen and Aktivitäten zur Verfügung.

### 11.4.1 Neue Prozessinstanz erzeugen

#### Initialisieren der Prozesse eines Objekts über SOAP

(FSCBAIWF@1.1001:SOAPInitializeWorkFlow)

Diese SOAP-Aktion initialisiert einen Workflow auf einem Fabasoft Folio Objekt. Es kann die gewünschte Prozessdefinition im Request-XML mitgegeben werden. Weiters besteht die Möglichkeit zu entscheiden, ob der neue Prozess zum alten hinzugefügt werden soll oder nicht. Hier kann es je nach Vorkonfiguration zu unterschiedlichen Ergebnissen kommen.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:baiwf:control

```
<SOAPInitializeWorkFlowRequest>
  <obj-id> Betroffenes Objekt
  ...
</obj-id>
<procdef-ids> Prozessdefinition(en)
...
</procdef-ids>
<additionalobj-ids> Weitere Objekte des Prozesses
...
</additionalobj-ids>
<submit-deadline> Termin für Vorlage </submit-deadline>
</SOAPInitializeWorkFlowRequest>
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:baiwf:control

```
<BAIWFNoResponse>
</BAIWFNoResponse>
```

---

Im Element `<obj-id>` wird die Objektadresse des Objekts, auf das der Workflow initialisiert werden soll, angegeben.

#### Beispiel:

In der Fabasoft Produktinstallation soll ein Prozess auf einen Objekt initialisiert werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Initialisieren der Prozesse eines Objektes über SOAP* (FSCBAI@1.1001:SOAPInitializeWorkFlow) übermittelt

```

::params=coort.CreateDictionary();
::returnval=coort.CreateDictionary();
::params=coort.CreateDictionary();
::objid=coort.CreateDictionary();
::procdef=coort.CreateDictionary();

::objid.SetEntryValue("id-ref",0,this.objaddress);
::params.SetEntryValue("obj-id",0,::objid);

::procdef.SetEntryValue("obj-id",0,"COO.10.531.1.10006");
::params.SetEntryValue("procdef-ids",0,::procdef);

::params.SetEntryValue("submit-deadline",0,"2007-12-12T12:00:00");

FSCBAI@1.1001:CallSoapXmlEx(::dept,"InitializeWorkFlow",::params,&::returnval)

```

## 11.4.2 Erstellen von Aktivitäten

### Externe Aktivität über SOAP erzeugen (FSCBAIWF@1.1001:SOAPCreateExternalActivity)

Mit dieser SOAP-Aktion können Objekte der Objektklasse *Externe Aktivität* (COOWF@1.1:ExternalActivityInstance) erstellt werden. Diese Aktivitäten müssen der SOAP-Aktion *Externe Aktivität über SOAP einfügen* (FSCBAIWF@1.1001:SOAPInsertExternalActivity) übergeben werden.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:baiwf:createactinst

```

<SOAPCreateExternalActivityRequest>
  <actinst-title> Name </actinst-title>
  <actinst-participant> Teilnehmer
  ...
</actinst-participant>
  <actinst-receivedat> Empfangen am </actinst-receivedat>
  <actinst-startedat> Gestartet am </actinst-startedat>
  <actinst-completedat> Beendet am </actinst-completedat>
</SOAPCreateExternalActivityRequest>

```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:baiwf:createactinst

```

<SOAPCreateExternalActivityResponseType>
  <result> Ergebnis der SOAP-Aktion </result>
  <actinst-id> Objektadresse der neuen Aktivität </actinst-id>
  <actinst-name> Name der neuen Aktivität </actinst-name>
</SOAPCreateExternalActivityResponseType>

```

#### Beispiel:

In der Fabasoft Produktinstallation soll eine neue *Externe Aktivität* erstellt werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über*

*Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Externe Aktivität über SOAP erzeugen* (FSCBAIWF@1.1001:SOAPCreateExternalActivity) übermittelt.

```
::params=coort.CreateDictionary();
::returnval=coort.CreateDictionary();
::participant=coort.CreateDictionary();
::idtype1=coort.CreateDictionary();
::idtype2=coort.CreateDictionary();

::idtype1.SetEntryValue("id-ref",0,"Adresse des User");
::idtype2.SetEntryValue("id-ref",0,"Adresse der Gruppe");
::participant.SetEntryValue("user-id",0,::idtype1);
::participant.SetEntryValue("group-id",0,::idtype2);
::params.SetEntryValue("actinst-title",0,"Test");
::params.SetEntryValue("actinst-participant",0,::participant);
::params.SetEntryValue("actinst-receivedat",0,"2005-02-02T12:30:00");
::params.SetEntryValue("actinst-startedat",0,"2005-02-02T12:10:00");
::params.SetEntryValue("actinst-completedat",0,"2005-02-02T12:20:00");
FSCBAI@1.1001:CallSoapXmlEx(::dept,"CreateExternalAct",::params,&::returnval);
```

### 11.4.3 Einfügen von Aktivitäten

**Externe Aktivität über SOAP einfügen** (FSCBAIWF@1.1001:SOAPInsertExternalActivity)

Diese SOAP-Aktion kann verwendet werden, um Aktivitäten der Objektklasse *Externe Aktivität* (COOWF@1.1:ExternalActivityInstance) in einen Prozess einzufügen.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:baiwf:control

```
<SOAPInsertExternalActivityRequest>
  <obj-id> Betroffenes Objekt mit Prozess im Wartezustand
  ...
</obj-id>
  <extinst-id> Einzufügende Aktivität der Objektklasse Externe Aktivität (diese
Aktivität muss vorher erzeugt werden)
  ...
</extinst-id>
  <ext-complete> Externe Bearbeitung fertig </ext-complete>
</SOAPInsertExternalActivityRequest>
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:baiwf:control

```
<BAIWFNoResponse>
</BAIWFNoResponse>
```

---

Ist bei einer Aktivitätsdefinition oder bei einer Aktivität die Eigenschaft *Aktivität wird von Extern synchronisiert* (actinstexternalsync bzw. actdefexternalsync) ausgewählt, so wird diese Aktivität (und folgende Aktivitäten) in einen Wartezustand versetzt. Ab diesem Zeitpunkt kann die Aktivität in der Liste *Externe Synchronisation* im Arbeitsvorrat eingesehen werden.

Während die Aktivität auf externe Synchronisation wartet, können mithilfe der angegebenen SOAP-Aktion Aktivitäten vor der Aktivität eingefügt werden. Die Fortsetzung des Prozesses muss explizit signalisiert werden, indem beim Aufruf der SOAP-Aktion das Element `<ext-complete>` auf `TRUE` gesetzt wird.

#### Beispiel:

In einem Prozess der Fabasoft Produktinstallation soll eine neue *Externe Aktivität* eingefügt werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Externe Aktivität über SOAP einfügen* (FSCBAIWF@1.1001:SOAPInsertExternalActivity) übermittelt.

```
::objid.SetEntryValue("id-ref",0,"Objektadresse");
::params.SetEntryValue("obj-id",0,::objid);
::extinstid.SetEntryValue("id-ref",0,„Objektadresse der Aktivität“);
::params.SetEntryValue("extinst-id",0,::extinstid);
::params.SetEntryValue("ext-complete",0,FALSE);
FSCBAI@1.1001:CallSoapXmlEx(,::dept,"InsertExternalActivity",,::params,&::returnval)
```

#### 11.4.4 Vorschreiben (FSCBAIWF@1.1001:SOAPPrescribeObject)

Mit dieser SOAP-Aktion kann einem Benutzer ein Ad-hoc-Workflow über SOAP vorgeschrieben werden. Der Benutzer kann pro vorgeschriebener Aktivität definiert werden.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:baiwf:control

```
<SOAPPrescribeObjectRequest>
  <obj-id> Betroffene Aktivitätsinstanz
  ...
</obj-id>
<remark> Bemerkung </remark>
<prescr-block>
  <prescr-parallel> Vorschreibungen parallel durchführen
  </prescr-parallel>
  <prescription> Vorgeschriebene Aktivitäten
  ...
</prescription>
</prescr-block>
  <notcomplete>Die alte Aktivität wird nicht beendet</notcomplete>
</SOAPPrescribeObjectRequest>
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:baiwf:control

```
<BAIWFNoResponse>
</BAIWFNoResponse>
```

Mit dieser Aktion können beliebige Aktivitäten vorgeschrieben werden. Ausgeführt wird die Aktion auf die Aktivitätsinstanz, die im Element `<obj-id>` angegeben wird. Wichtig ist, dass diese Aktivität den Status „Begonnen“ hat. Ansonsten schlägt diese SOAP-Aktion fehl.

### Beispiel:

In einem Prozess der Fabasoft Produktinstallation soll eine Aktivität vorgeschrieben werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Objekt vorschreiben* (FSCBAIWF@1.1001:SOAPPrescribeObject) übermittelt.

```
::params=coort.CreateDictionary();
::returnval=coort.CreateDictionary();
::objid=coort.CreateDictionary();
::prescrblock=coort.CreateDictionary();
::prescription=coort.CreateDictionary();
::participant=coort.CreateDictionary();
::userid=coort.CreateDictionary();
::objid.SetEntryValue("id-ref",0,"Adresse der Aktivität");
::params2.SetEntryValue("obj-id",0,::objid);
::actdefprescriid.SetEntryValue("id-ref",0,"Objektadresse der Aktivität, die vorgeschrieben werden soll ");
::userid.SetEntryValue("id-ref",0,"Teilnehmer");
::participant.SetEntryValue("user-id",0,::userid);
::prescription.SetEntryValue("actdefprescr-id",0,::actdefprescriid);
::prescription.SetEntryValue("participant",0,::participant);
::prescription.SetEntryValue("remark",0,"Test");
::prescription.SetEntryValue("actdefback-id",0,::actdefbackid);
::prescrblock.SetEntryValue("prescription",0,::prescription);
::params.SetEntryValue("remark",0,"Test");
::params.SetEntryValue("prescr-block",0,::prescrblock);
::params.SetEntryValue("notcomplete",0,"FALSE");
FSCBAI@1.1001:CallSoapXmlEx(::dept,"PrescribeObject",::params,&::returnval)
```

Bemerkungen können sowohl bei der vorgeschriebenen Aktivität, als auch bei den neuen Aktivitäten eingegeben werden. Wichtig ist, dass die allgemeine Bemerkung nur dann eingetragen wird, wenn der XML-Knoten `<notcomplete>` auf `FALSE` gesetzt wird.

## 11.4.5 Zuteilen

**Zustand der Aktivität auf 'zuteilt' setzen** (FSCBAIWF@1.1001:SOAPSetDelegated)

Mit dieser Aktion kann eine Aktivität einem Workflowteilnehmer zuteilt werden.

---

### Input

---

**Namespace:** urn:schemas-fabasoft-com:baiwf:control

```
<SOAPSetDelegatedRequest>
  <obj-id> Betroffene Aktivitätsinstanz
  ...
  </obj-id>
  <participant> Teilnehmer
  ...
  </participant>
  <partmansubst> Manueller Teilnehmer
```

---

```
...
</partmansubst>
<remark> Bemerkung </remark>
<submit-deadline> Termin für Vorlage </submit-deadline>
<start-deadline> Termin für Beginn </start-deadline>
<end-deadline> Termin für Erledigung </end-deadline>
</SOAPSetDelegatedRequest>
```

---

## Output

---

**Namespace:** urn:schemas-fabasoftware-com:baiwf:control

```
<BAIWFNoResponse>
</BAIWFNoResponse>
```

Mit dieser Aktion wird die angegebene Aktivitätsinstanz einem in `<participant>` bzw. `<partmansubst>` angegebenen Workflowteilnehmer zugeteilt. Wichtig ist, dass die Aktivität, die im Knoten `<obj-id>` angegeben ist, den Status „Begonnen“ hat.

### Beispiel:

In einem Prozess der Fabasoft Produktinstallation soll eine Aktivität zugeteilt werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Zustand der Aktivität auf „zugeteilt“ setzen* (FSCBAIWF@1.1001:SOAPSetDelegated) übermittelt.

```
::params=coort.CreateDictionary();
::returnval=coort.CreateDictionary();
::objid=coort.CreateDictionary();
::participant=coort.CreateDictionary();
::userid=coort.CreateDictionary();
::groupid=coort.CreateDictionary();

::objid.SetEntryValue("id-ref",0,"Aktivitätsinstanz");
::params.SetEntryValue("obj-id",0,::objid);

::participant.SetEntryValue("meta-participant",0,0);
::userid.SetEntryValue("id-ref",0,"Adresse des Users");
::participant.SetEntryValue("user-id",0,::userid);
::groupid.SetEntryValue("id-ref",0,"Adresse der Gruppe");
::participant.SetEntryValue("group-id",0,::groupid);
::params.SetEntryValue("participant",0,::participant);
::params.SetEntryValue("remark",0,"Test");
::params.SetEntryValue("submit-deadline",0,"Datum");
::params.SetEntryValue("start-deadline",0,"Datum");
::params.SetEntryValue("end-deadline",0,"Datum");

FSCBAI@1.1001:CallSoapXmlEx(::dept,"SetDelegated",::params,&::returnval)
```

## 11.4.6 Weiterleiten/Beenden einer Aktivität (FSCBAIWF@1.1001:SOAPSetCompleted)

Soll eine Aktivität, die den Status „Begonnen“ hat über SOAP weitergeleitet bzw. beendet werden, muss diese SOAP-Aktion aufgerufen werden.

---

### Input

**Namespace:** urn:schemas-fabasoftware-com:baiwf:control

```
<SOAPSetCompletedRequest>
  <obj-id> Betroffene Aktivitätsinstanz
  ...
</obj-id>
  <remark> Bemerkung </remark>
</SOAPSetCompletedRequest>
```

---

### Output

**Namespace:** urn:schemas-fabasoftware-com:baiwf:control

```
<BAIWFNoResponse>
</BAIWFNoResponse>
```

---

Im Element `<obj-id>` wird die Objektadresse der Aktivitätsdefinition angegeben, die beendet werden soll. Die Aktivität muss den Status „Begonnen“ haben.

#### Beispiel:

In einem Prozess der Fabasoft Produktinstallation soll eine Aktivität weitergeleitet werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Weiterleiten/Beenden einer Aktivität* (FSCBAIWF@1.1001:SOAPSetCompleted) übermittelt.

```
::params=coort.CreateDictionary();
::returnval=coort.CreateDictionary();
::objid.SetEntryValue("id-ref",0,"Adresse der Aktivität");
::params.SetEntryValue("obj-id",0,::objid);
::params.SetEntryValue("remark",0,"Test");
```

```
FSCBAI@1.1001:CallSoapXmlEx( :dept, "SetCompleted", :params, &::returnval)
```

## 11.4.7 Unterschreiben

Die Einstellungen einer *Unterschriftenart* (COOSIGNATURE@1.1:SignatureType) können mittels der Eigenschaft *Änderungen der vordefinierten Eigenschaften* (COOSIGNATURE@1.1:sgtmodifications) je Mandant überschrieben werden. Diese Änderungen können dynamisch via Expression (COOSIGNATURE@1.1:sgmcondition) deaktiviert werden, damit z.B. mittels einer Transaktionsvariable die Einstellungen einer Unterschriftenart (wie z.B. *Art der Authentisierung*) geändert werden können.

Global Scope	DICTIONARY
signtype	Unterschriftenart
modification	Änderungen der vordefinierten Eigenschaften
Local Scope	DICTIONARY
(empty)	
Result	BOOLEAN

## 11.5 Lösungsbereich Suche

Die Fabasoft Produktinstallation bietet einer Fachanwendung die Möglichkeit Objekte zu suchen. Die Ergebnisse werden aufbereitet zurückgeliefert.

### 11.5.1 Objekte via XML/SOAP suchen und Eigenschaften auslesen

Die SOAP-Aktion *Objekte via XML/SOAP suchen* und Eigenschaften auslesen (FSCBAI@1.1001:SOAPSearch) bietet die Möglichkeit, Objekte über eine Standard-Suchabfrage zu suchen und die Metadaten auszulesen.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:bai:search

```
<SOAPSearchRequest>
  <query>
    Fabasoft Folio Suchabfrage
  </query>
  <mapping> Objektadresse des Mappings </mapping>
  <dept> Referenz des Fachanwendungsbereichs </dept>
</SOAPSearchRequest>
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:bai:search

```
<SOAPSearchResponse>
  <queryresult>
    <object> //n-mal
      <objname> Objektname </objname>
      <objaddress> Objektadresse </objaddress>
      <data> Ergebnis-XML laut Mapping </data>
      ...
    </object>
  </queryresult>
</SOAPSearchResponse>
```

Die Aktion erwartet im XML-Element `<query>` eine Fabasoft Folio Suchabfrage. Für die gefundenen Objekte werden Basisinformationen (Objektname im Element `<objname>` und Objektadresse im Element `<objaddress>`) zurückgeliefert.

Wird im SOAP-Request für den Aufruf im XML-Knoten `mapping` eine XML-Abbildung für die Objektklasse der gesuchten Objekte übergeben, so wird diese Abbildung auf jedes Objekt der Suchergebnisliste angewendet. Das daraus generierte XML-Dokument wird an die Stelle des `<any>`-Elements im Output-XML-Dokument der Aktion eingefügt.

Weiters besteht die Möglichkeit das Mapping über die Konfiguration für Fachanwendungen zu ermitteln. Hier muss wieder pro Objektklasse, Fachanwendungsbereich und dem Auslöser `FSCBAI@1.1001:SOAPSearch` ein Mapping für die gewünschte Objektklasse eingetragen werden.

Wird im SOAP-Request für den Aufruf im XML-Knoten `attrlist` ein oder mehrere XML-Knoten `attr` mit einer Fabasoft DUCX Expression übergeben, so werden diese auf jedes Objekt der Suchergebnisliste angewendet. Das Ergebnis wird in einer, dem Datentyp des Ergebnisses angelehnten, XML-Struktur abgebildet und an die Stelle des `<attrlist>`-Elements im Output-XML-Dokument der Aktion eingefügt.

### Beispiel:

Ausschnitt aus dem SOAP-Request:

```
<attrlist>
  <attr>COOSYSTEM@1.1:objcreatedby</attr>
  <attr>COOSYSTEM@1.1:content</attr>
</attrlist>
```

Ausschnitt aus dem SOAP-Response:

```
<queryresult>
  <object>
    <objname>Search Note</objname>
    <objaddress>COO.1.1065.3.3008348</objaddress>
    <attrlist>
      <fsc1:OBJECT xmlns:fsc1="urn:schemas-fabasoft-com:bai:search"
        fsc1:reference="COOSYSTEM@1.1:objcreatedby"
        fsc1:index="0">COO.1.1065.1.16</fsc1:OBJECT>
      <fsc1:AGGREGATE xmlns:fsc1="urn:schemas-fabasoft-com:bai:search"
        fsc1:reference="COOSYSTEM@1.1:content" fsc1:index="0">
        <fsc1:CONTENT fsc1:reference="COOSYSTEM@1.1:contcontent"
          fsc1:index="0">WW91IGNhbiBkZWVvZGUgQmFzZTY0IQ==</fsc1:CONTENT>
        <fsc1:INTEGER fsc1:reference="COOSYSTEM@1.1:contsize"
          fsc1:index="0">22</fsc1:INTEGER>
        <fsc1:DATETIME fsc1:reference="COOSYSTEM@1.1:contchanged"
          fsc1:index="0">2009-09-18 12:04:22</fsc1:DATETIME>
        <fsc1:STRING fsc1:reference="COOSYSTEM@1.1:contextension"
          fsc1:index="0">txt</fsc1:STRING>
      </fsc1:AGGREGATE>
    </attrlist>
  </object>
</queryresult>
```

Falls beim Aufruf der Aktion ein Fehler auftritt, wird ein Standard-SOAP-Fehler generiert.

**Hinweis:** Der Aufbau einer gültigen Fabasoft Folio Suchabfrage kann der Entwicklerdokumentation von Fabasoft Folio entnommen werden.

### Beispiel:

Es soll in der Fabasoft Produktinstallation nach Inhaltsobjekten gesucht werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Objekte via XML/SOAP suchen und Eigenschaften auslesen* (FSCBAI@1.1001:SOAPSearch) übermittelt.

```
::returnval=coort.CreateDictionary();
::params =coort.CreateDictionary();
::params.SetEntryValue("query",0, „LIMIT 100 SELECT * FROM
COOSYSTEM@1.1:ContentObject“);
::params.SetEntryValue("mapping",0, „COO-Adresse des Mappings“);
FSCBAI@1.1001:CallsSoapXmlEx(::dept,„SOAPSearch“,::params,&::returnval)
```

## 11.6 Lösungsbereich Check-out - Check-in

Werden Änderungen eines Fabasoft Folio Objekts von der Fachanwendung durchgeführt, kann diese das betreffende Objekt in der Fabasoft Produktinstallation sperren und wieder entsperren. So wird verhindert, dass zur gleichen Zeit ein anderer Benutzer Änderungen an diesem Objekt vornimmt.

### 11.6.1 Auschecken über SOAP (FSCBAI@1.1001:SOAPCheckOut)

Ein Check-out eines Objekts sperrt das betreffende Objekt in der Fabasoft Produktinstallation (permanente Sperre). Darüber hinaus können weitere Objekte mitgesperrt werden. Bei allen gesperrten Objekten wird eine neue Version begonnen.

---

#### Input

**Namespace:** urn:schemas-fabasoft-com:SOAPCheckInOut

```
<SOAPCheckOutRequest>
  <id> Objektadresse </id>
  <lock> Objekte permanent sperren Ja/Nein </lock>
  <lockpropagate> Sperren weiterführen Ja/Nein
</lockpropagate>
  <dept> Fachanwendungsbereich </dept>
</SOAPCheckOutRequest>
```

---

#### Output

**Namespace:** urn:schemas-fabasoft-com:SOAPCheckInOut

```
<SOAPCheckOutResponse>
  <idcode> Zeichenkette mit Statusinformationen </idcode>
  <doc> XML-Daten entsprechend dem Mapping
  ...
</doc>
  <lockfailed> Nicht gesperrte Objekte
  <object> Objektadresse </object>
  ...
</lockfailed>
  <locksucces> Gesperrte Objekte
```

---

```

    <object> Objektadresse </object>
    ...
  </locksuccess>
</SOAPCheckOutResponse>

```

Das Element `<lockpropagate>` gibt an, ob die Sperre auf alle Objekte weitergeführt werden soll, die durch einen Ausdruck in der Eigenschaft *Weiterführen von permanenten Sperren* (FSCBAI@1.1001:propagatedlocks) in der *Konfiguration für Fachanwendungen* angegeben sind. Wenn im Element `<lock>` `FALSE` angegeben wird, wird das Element `<lockpropagate>` nicht berücksichtigt.

Im Element `<idcode>` wird eine Zeichenkette zurückgegeben, die beim Check-in-Aufruf wieder mitgegeben werden muss, um alle Objekte wieder entsperren zu können, die beim Check-out gesperrt wurden. In `<lockfailed>` werden die Objektadressen der Objekte zurückgegeben, bei denen versucht wurde eine permanente Sperre zu setzen, dabei jedoch ein Fehler aufgetreten ist (z. B. durch fehlende Berechtigungen etc.). In `<locksuccess>` werden die Objektadressen der Objekte zurückgegeben, bei denen eine permanente Sperre gesetzt wurde.

Das Element `<doc>` ist vom Typ „any“ und beinhaltet die ausgecheckten Daten. Das Schema für diese Daten kann in der *Konfiguration für Fachanwendungen* in der Eigenschaft *Schematazuordnungen* (FSCBAI@1.1001:schemamaps) definiert werden. Die gewünschte Schematazuordnung muss dem *Fachanwendungsbereich* zugewiesen werden, der auch der SOAP-Aktion mitgegeben wurde. Ist das nicht der Fall, kann keine Transformation durchgeführt werden.

### Beispiel:

Es soll ein Objekt mit Objektliste ausgecheckt werden. Die weiterführenden Sperren müssen aktiviert sein. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Auschecken über SOAP* (FSCBAI@1.1001:SOAPCheckOut) übermittelt.

```

::returnval=coort.CreateDictionary();
::params.SetEntryValue("id",0,this.objaddress);
::params.SetEntryValue("lock",0,"TRUE");
::params.SetEntryValue("lockpropagate",0,"TRUE");
::params.SetEntryValue("dept",0,::dept.objaddress);
FSCBAI@1.1001:CallSoapXmlEx(::dept,"CheckOut",::params,&::returnval);
objsubject=::returnval.GetEntryValue2("idcode",0)

```

## 11.6.2 Einchecken über SOAP (FSCBAI@1.1001:SOAPCheckIn)

Beim Check-in werden die Daten aktualisiert und anschließend wird optional das Objekt und gegebenenfalls weitere Objekte, die beim Check-out gesperrt wurden, wieder freigegeben. Zusätzlich wird eine neue Version begonnen.

### Input

**Namespace:** urn:schemas-fabasoft-com:SOAPCheckInOut

```

<SOAPCheckInRequest>
  <idcode> Zeichenkette mit Statusinformationen </idcode>
  <overwrite> Überschreiben, falls Objekt geändert Ja/Nein

```

```

</overwrite>
<unlock> Objekte entsperren Ja/Nein </unlock>
<doc> XML-Daten entsprechend dem Mapping
...
</doc>
<dept> Fachanwendungsbereich </dept>
</SOAPCheckInRequest>

```

---

## Output

---

**Namespace:** urn:schemas-fabasoft-com:SOAPCheckInOut

```

<SOAPCheckInResponse>
  <resultstring> Ergebnis der SOAP-Aktion </resultstring>
</SOAPCheckInResponse>

```

Im Element `<idcode>` muss die Zeichenkette übergeben werden, die beim Check-out-Aufruf zurückgegeben wurde, um alle beim Check-out gesperrten Objekte wieder entsperren zu können. In `<overwrite>` wird angegeben, ob die Objekte auch dann überschrieben werden sollen, wenn die Objekte in der Zwischenzeit geändert wurden. Die Änderungen werden anhand des Änderungsdatums und des Erstellungsdatums der aktuellen Version erkannt. Standardwert für `<overwrite>` ist `FALSE`. Das Element `<unlock>` gibt an, ob die Objekte nach dem Check-in wieder entsperrt werden sollen. Dadurch ist es möglich, z. B. zwischendurch mehrmals Daten zurück zu schreiben ohne dabei die Objekte zu entsperren. Der Standardwert für `<unlock>` ist `TRUE`. Fehler werden über den SOAP-Fehlerstatus zurückgegeben.

Das Element `<doc>` ist vom Typ „any“ und beinhaltet die Daten, die beim Einchecken in das Geschäftsobjekt geschrieben werden sollen. Das Schema für diese Daten wird in der *Konfiguration für Fachanwendungen* in der Eigenschaft *Schematazuordnungen* (FSCBAI@1.1001:schemamaps) aus der dort angegebenen *Abbildung von XML-Elementen auf Objekteigenschaften* ausgelesen. Damit das XML-Schema in der Konfiguration für Fachanwendungen gefunden werden kann, muss auch der entsprechende *Fachanwendungsbereich* im XML-Knoten `<dept>` übergeben werden.

### Beispiel:

Es soll ein Objekt mit Objektliste eingechekkt werden. Die permanenten Sperren müssen deaktiviert sein. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Einchecken über SOAP* (FSCBAI@1.1001:SOAPCheckIn) übermittelt.

```

::params=coort.CreateDictionary();
::returnval=coort.CreateDictionary();
::params.SetEntryValue("idcode",0,objsubject);
::params.SetEntryValue("overwrite",0,"TRUE");
::params.SetEntryValue("unlock",0,TRUE);
::params.SetEntryValue("dept",0,::dept.objaddress);
FSCBAI@1.1001:CallSoapXmlEx(::dept,"CheckIn",::params,&::returnval);

```

## 11.7 Generische SOAP-Aktionen

Im Folgenden finden Sie generische SOAP-Aktionen.

### 11.7.1 Generisches Lesen von Eigenschaften von Objekten

Die SOAP-Aktion *Eigenschaften über XML/SOAP lesen (generisch)*

(FSCBAI@1.1001:SOAPGenericGetProperties) bietet die Möglichkeit, Eigenschaften eines Objekts generisch auszulesen.

---

#### Input

---

**Namespace:** urn:schemas-fabasoftware-com:universal

```
<object>
  <objaddress> Objektadresse des Objekts
</objaddress>
  <attrlist>
    <attr> Referenz einer Fabasoft Folio Eigenschaft
    </attr>
    ...
  </attrlist>
</object>
```

---

#### Output

---

**Namespace:** urn:schemas-fabasoftware-com:universal

```
<OBJECT objaddress="Objektadresse des Objekts">
  <BOOLEAN reference="Referenz der Eigenschaft"> Wert der Eigenschaft </BOOLEAN>
  <BOOLEANLIST reference="Referenz der Eigenschaft">
    <BOOLEAN reference="..."> Wert der Eigenschaft </BOOLEAN>
    ...
  </BOOLEANLIST>
  <STRING reference="..."> Wert der Eigenschaft </STRING>
  <STRINGLIST reference="...">
    <STRING reference="..."> Wert der Eigenschaft </STRING>
    ...
  </STRINGLIST>
  <FLOAT reference="..."> Wert der Eigenschaft </FLOAT>
  <FLOATLIST reference="...">
    <FLOAT reference="..."> Wert der Eigenschaft </FLOAT>
    ...
  </FLOATLIST>
  <INTEGER reference="..."> Wert der Eigenschaft </INTEGER>
  <INTEGERLIST reference="...">
    <INTEGER reference="..."> Wert der Eigenschaft </INTEGER>
    ...
  </INTEGERLIST>
  <CONTENT reference="..."> Wert der Eigenschaft </CONTENT>
  <CONTENTLIST reference="...">
    <CONTENT reference="..."> Wert der Eigenschaft </CONTENT>
    ...
  </CONTENTLIST>
  <OBJECT reference="..."> Wert der Eigenschaft </OBJECT>
  <OBJECTLIST reference="...">
    <OBJECT reference="..."> Wert der Eigenschaft </OBJECT>
    ...
  </OBJECTLIST>
```

---

```

<DATETIME reference="..."> Wert der Eigenschaft </DATETIME>
<DATETIMELIST reference="...">
  <DATETIME reference="..."> Wert der Eigenschaft </DATETIME>
  ...
</DATETIMELIST>
<ENUM reference="..."> Wert der Eigenschaft </ENUM>
<ENUMLIST reference="...">
  <ENUM reference="..."> Wert der Eigenschaft </ENUM>
  ...
</ENUMLIST>
<AGGREGATE reference="..."> Wert der Eigenschaft </AGGREGATE>
<AGGREGATELIST reference="...">
  <AGGREGATE reference="..."> Wert der Eigenschaft </AGGREGATE>
  ...
</AGGREGATELIST>
</OBJECT>

```

Die Aktion liefert zu dem im Element `<objaddress>` angegebenen Objekt die Werte jener Eigenschaften, die in `<attrlist>` angegeben sind. Falls keine Eigenschaften spezifiziert sind, werden alle Eigenschaften des Objekts zurückgeliefert.

Um die Antwort zu erzeugen werden die Werte der Eigenschaften entsprechend ihres Typs in Skalare und Listen gruppiert. Das XML-Attribut `reference` speichert die Referenz der Eigenschaft, wodurch die Werte eindeutig zugeordnet werden können.

Die Werte der Fabasoft Folio Eigenschaften werden wie folgt auf die XML-Elemente entsprechend den XML-Schema-Datentypen (XSD) abgebildet:

Datentyp	Beschreibung
<BOOLEAN>	Wert einer booleschen Eigenschaft, 0 für FALSE, 1 für TRUE
<BOOLEANLIST>	Liste von booleschen Eigenschaften, wobei jedes Element der Liste in einem <BOOLEAN>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<STRING>	Wert einer Zeichenketteneigenschaft, UTF-8-Zeichenkette
<STRINGLIST>	Liste von Zeichenketteneigenschaften, wobei jedes Element der Liste in einem <STRING>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<FLOAT>	Wert einer Gleitkommaeigenschaft
<FLOATLIST>	Liste von Gleitkommaeigenschaften, wobei jedes Element der Liste in einem <FLOAT>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<INTEGER>	Wert einer ganzzahligen Eigenschaft

<INTEGERLIST>	Liste von ganzzahligen Eigenschaften, wobei jedes Element der Liste in einem <INTEGER>-XML-Element ohne <i>reference</i> -Attribut angegeben ist.
<CONTENT>	Wert einer Eigenschaft für Inhalte, Base64-kodierter Inhalt
<CONTENTLIST>	Liste von Inhaltseigenschaften, wobei jedes Element der Liste in einem <CONTENT>-XML-Element ohne <i>reference</i> -Attribut angegeben ist.
<OBJECT>	Wert einer Objektzeigereigenschaft, Objektadresse
<OBJECTLIST>	Liste von Objektzeigereigenschaften, wobei jedes Element der Liste in einem <OBJECT>-XML-Element ohne <i>reference</i> -Attribut angegeben ist.
<DATETIME>	Wert einer Eigenschaft für Datum/Zeit
<DATETIMELIST>	Liste von Eigenschaften für Datum/Zeit, wobei jedes Element der Liste in einem <DATETIME>-XML-Element ohne <i>reference</i> -Attribut angegeben ist.
<ENUM>	Wert einer Aufzählungseigenschaft, Zahlenwert
<ENUMLIST>	Liste von Aufzählungseigenschaften, wobei jedes Element der Liste in einem <ENUM>-XML-Element ohne <i>reference</i> -Attribut angegeben ist.
<AGGREGATE>	Wert einer zusammengesetzten Eigenschaft, der wieder aus den Elementen <BOOLEAN>, <STRING>, <FLOAT>, <INTEGER>, <CONTENT>, <OBJECT>, <DATETIME>, <ENUM> und <AGGREGATE> bestehen kann.
<AGGREGATELIST>	Liste von zusammengesetzten Eigenschaften, wobei jedes Element der Liste in einem <AGGREGATE>-XML-Element ohne <i>reference</i> -Attribut angegeben ist.

### Beispiel:

Anhand der definierten Eigenschaften sollen Metadaten eines Objekts gelesen werden. Der XML-Request wird mithilfe einer Fabasoft DUCX Expression erstellt und unter Verwendung der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) an die SOAP-Aktion *Eigenschaften über XML/SOAP lesen (generisch)* (FSCBAI@1.1001:SOAPGenericGetProperties) übermittelt.

```
::returnval=coort.CreateContent();
::params=coort.CreateDictionary();
::attrlist=coort.CreateDictionary();

::params.SetEntryValue("objaddress",0,this.objaddress);
```

```

::attrlist.SetEntryValue("attr",0,"COOSYSTEM@1.1:content.COOSYSTEM@1.1:contsize");
::attrlist.SetEntryValue("attr",1,"COOSYSTEM@1.1:objcreatedat");
::attrlist.SetEntryValue("attr",2,"COOSYSTEM@1.1:objcreatedby");
::params.SetEntryValue("attrlist",0,::attrlist);

```

```
FSCBAI@1.1001:CallSoapXmlEx(,::dept,"GenericGetProperties",,::params,&::returnval)
```

## 11.7.2 Generisches Ändern von Eigenschaften von Objekten

Die SOAP-Aktion *Eigenschaften über XML/SOAP setzen (generisch)*

(FSCBAI@1.1001:SOAPGenericSetProperties) bietet die Möglichkeit, Eigenschaften eines Objekts zu ändern.

---

### Input

**Namespace:** urn:schemas-fabasoftware-com:universal

```

<OBJECT objaddress="Objektadresse des Objekts"
  create="Referenz einer Objektklasse">
  <BOOLEAN reference="Referenz der Eigenschaft"> Wert der Eigenschaft </BOOLEAN>
  <BOOLEANLIST reference="Referenz der Eigenschaft">
    <BOOLEAN reference="..."> Wert der Eigenschaft </BOOLEAN>
    ...
  </BOOLEANLIST>
  <STRING reference="..."> Wert der Eigenschaft </STRING>
  <STRINGLIST reference="...">
    <STRING reference="..."> Wert der Eigenschaft </STRING>
    ...
  </STRINGLIST>
  <FLOAT reference="..."> Wert der Eigenschaft </FLOAT>
  <FLOATLIST reference="...">
    <FLOAT reference="..."> Wert der Eigenschaft </FLOAT>
    ...
  </FLOATLIST>
  <INTEGER reference="..."> Wert der Eigenschaft </INTEGER>
  <INTEGERLIST reference="...">
    <INTEGER reference="..."> Wert der Eigenschaft </INTEGER>
    ...
  </INTEGERLIST>
  <CONTENT reference="..."> Wert der Eigenschaft </CONTENT>
  <CONTENTLIST reference="...">
    <CONTENT reference="..."> Wert der Eigenschaft </CONTENT>
    ...
  </CONTENTLIST>
  <OBJECT reference="..."> Wert der Eigenschaft </OBJECT>
  <OBJECTLIST reference="...">
    <OBJECT reference="..."> Wert der Eigenschaft </OBJECT>
    ...
  </OBJECTLIST>
  <DATETIME reference="..."> Wert der Eigenschaft </DATETIME>
  <DATETIMELIST reference="...">
    <DATETIME reference="..."> Wert der Eigenschaft </DATETIME>
    ...
  </DATETIMELIST>
  <ENUM reference="..."> Wert der Eigenschaft </ENUM>
  <ENUMLIST reference="...">
    <ENUM reference="..."> Wert der Eigenschaft </ENUM>
    ...
  </ENUMLIST>
  <AGGREGATE reference="..."> Wert der Eigenschaft </AGGREGATE>
  <AGGREGATELIST reference="...">
    <AGGREGATE reference="..."> Wert der Eigenschaft </AGGREGATE>
    ...
  </AGGREGATELIST>
  ...
</OBJECT>

```

```
</AGGREGATELIST>
</OBJECT>
```

---

## Output

---

**Namespace:** urn:schemas-fabasoftware-com:universal

```
<response>
  <object> Objektadresse des geänderten/erzeugten Objekts
</object>
  <result> Ergebnis der SOAP-Aktion </result>
  <failed>
    <attr>
      Referenz der Eigenschaft, die nicht geändert wurde
    </attr>
    ...
  </failed>
</response>
```

---

Die Aktion ermittelt das zu ändernde Objekt anhand des XML-Attributs `objaddress`, das eine Objektadresse enthält. Falls im XML-Attribut `create` die Referenz einer Objektklasse angegeben ist, wird das zu ändernde Objekt nicht gesucht sondern neu erzeugt.

Anschließend werden die XML-Daten gruppiert nach ihren Fabasoft Folio Typen in die entsprechenden Eigenschaften des Objekts geladen. Über die *Referenz* der Eigenschaft und den Typ kann eine Eigenschaft eindeutig identifiziert werden.

Die Aktion liefert Informationen über das geänderte Objekt, sowie eine Liste jener Eigenschaften, die nicht geändert werden konnten. Zusätzlich wird der Ergebniswert der Aktion im XML-Element `<result>` zurückgegeben.

Die Werte der Fabasoft Folio Eigenschaften müssen wie folgt auf die XML-Elemente entsprechend den XML-Schema-Datentypen (XSD) abgebildet werden:

---

Datentyp	Beschreibung
<BOOLEAN>	Wert einer booleschen Eigenschaft, 0 für FALSE, 1 für TRUE
<BOOLEANLIST>	Liste von booleschen Eigenschaften, wobei jedes Element der Liste in einem <BOOLEAN>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<STRING>	Wert einer Zeichenketteneigenschaft, UTF-8-Zeichenkette
<STRINGLIST>	Liste von Zeichenketteneigenschaften, wobei jedes Element der Liste in einem <STRING>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<FLOAT>	Wert einer Gleitkommeeigenschaft

---

<FLOATLIST>	Liste von Gleitkommawerten, wobei jedes Element der Liste in einem <FLOAT>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<INTEGER>	Wert einer ganzzahligen Eigenschaft
<INTEGERLIST>	Liste von ganzzahligen Eigenschaften, wobei jedes Element der Liste in einem <INTEGER>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<CONTENT>	Wert einer Eigenschaft für Inhalte, Base64-kodierter Inhalt
<CONTENTLIST>	Liste von Inhaltseigenschaften, wobei jedes Element der Liste in einem <CONTENT>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<OBJECT>	Wert einer Objektzeigereigenschaft, Objektadresse
<OBJECTLIST>	Liste von Objektzeigereigenschaften, wobei jedes Element der Liste in einem <OBJECT>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<DATETIME>	Wert einer Eigenschaft für Datum/Zeit
<DATETIMELIST>	Liste von Eigenschaften für Datum/Zeit, wobei jedes Element der Liste in einem <DATETIME>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<ENUM>	Wert einer Aufzählungseigenschaft, Zahlenwert
<ENUMLIST>	Liste von Aufzählungseigenschaften, wobei jedes Element der Liste in einem <ENUM>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.
<AGGREGATE>	Wert einer zusammengesetzten Eigenschaft, der der rekursiven Natur dieser Eigenschaften entsprechend wieder aus den Elementen <BOOLEAN>, <STRING>, <FLOAT>, <INTEGER>, <CONTENT>, <OBJECT>, <DATETIME>, <ENUM> und <AGGREGATE> bestehen kann.
<AGGREGATELIST>	Liste von zusammengesetzten Eigenschaften, wobei jedes Element der Liste in einem <AGGREGATE>-XML-Element ohne <code>reference</code> -Attribut angegeben ist.

### Beispiel:

In der Fabasoft Produktinstallation sollen die Metadaten eines Objekts generisch gesetzt werden. Der XML-Request beinhaltet einen XML-Inhalt und wird als Parameter der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich*

(FSCBAI@1.1001:CallSoapXmlEx) übergeben. Diese ruft wiederum die SOAP-Aktion *Eigenschaften über XML/SOAP setzen (generisch)* (FSCBAI@1.1001:SOAPGenericSetProperties) auf.

Beispiel eines Request-XML:

```
<ns1:OBJECT ns1:objaddress="COO.20.75.4.191"
  xmlns:ns1="urn:schemas-fabasoftware-com:universal"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ns1:STRING ns1:reference=COOSYSTEM@1.1:objname
    dt:dt="string">Das ist ein Test</ns1:STRING>
  <ns1:AGGREGATELIST
    ns1:reference="COOMAPI@1.1:emailinformation">
  <ns1:AGGREGATE>
    <ns1:BOOLEAN ns1:reference=COOMAPI@1.1:emailrtf
      dt:dt="boolean">1</ns1:BOOLEAN>
    <ns1:STRING ns1:reference=COOMAPI@1.1:emailaddress
      dt:dt="string">Das ist ein SOAP-Test</ns1:STRING>
    <ns1:ENUM ns1:reference="COOMAPI@1.1:emailknowntype"
      dt:dt="int">2</ns1:ENUM>
  </ns1:AGGREGATE>
  </ns1:AGGREGATELIST>
</ns1:OBJECT>
```

Dieses XML muss als Inhalt in der Fabasoft DUCX Expression der Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) übergeben werden.

```
::returnval=coort.CreateDictionary();
::xmlcontent="Obiger Content";
FSCBAI@1.1001:CallSoapXmlEx(::dept,"GenericSetProperties",::xmlcontent,&::returnval
)
```

## 12 Funktionsteilbereich „Aufruf von SOAP-Aktionen einer Fachanwendung durch die Fabasoft Produktinstallation“

Die Fabasoft Produktinstallation bietet Funktionalität, um SOAP-Aktionen einer Fachanwendung aufzurufen.

### 12.1 Lösungsbereich „Nutzung von Funktionalität einer Fachanwendung“

#### 12.1.1 Standardinitialisierung eines Objekts über SOAP

Der Aufruf der SOAP-Aktion *Erzeugen eines Fachobjekts in Fabasoft Folio über XML/SOAP* (FSCBA@1.1001:SOAPInit), die von der Fachanwendung zur Verfügung gestellt wird, erfolgt entweder durch die Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (FSCBAI@1.1001:CallSoapXmlEx) oder durch die speziell dafür vorgesehene Aktion *Standardinitialisierung eines Objekts über SOAP* (FSCBAI@1.1001:InitBA) mit folgendem Parameter:

Parametername	IN/OUT	Typ	Erklärung
---------------	--------	-----	-----------

---

dept	In	OBJECT	Ein Objekt der Objektklasse <i>Fachanwendungsbereichsdefinition</i>
------	----	--------	--

---

Durch die Aktion `FSCBAI@1.1001:InitBA` wird nach der Antwort durch die SOAP-Aktion der Fachanwendung der Fremdschlüssel aus der Fachanwendung im Geschäftsobjekt (im Fachanwendungsaggregat) in der Fabasoft Produktinstallation gespeichert. Die Aktion führt einen SOAP-Aufruf entsprechend dem Schema der SOAP-Aktion `FSCBA@1.1001:SOAPInit` durch.

---

## Input

---

**Namespace:** urn:schemas-fabasoft-com:bai:init

```
<SOAPInitRequest>
  <objaddress> Objektadresse des Geschäftsobjekts
</objaddress>
  <deptref> Eindeutiger Bezeichner des Fachanwendungsbereichs
</deptref>
  <objname> Objektname des Geschäftsobjekts </objname>
  <objclass> Objektklasse in der Fachanwendung </objclass>
</SOAPInitRequest>
```

---

## Output

---

**Namespace:** urn:schemas-fabasoft-com:bai:init

```
<SOAPInitResponse>
  <foreignkey> Fremdschlüssel des Fachobjekts </foreignkey>
</SOAPInitResponse>
```

---

Das Element `<objclass>` wird mit der Zeichenkette aus der Eigenschaft *Objektyp für Standardinitialisierung im Fachanwendungsbereich* aus dem Fachanwendungsbereichsdefinitionsobjekt befüllt.

### Beispiel:

Die SOAP-Aktion kann z. B. in der *Konfiguration für Fachanwendungen* in der Eigenschaft *Fachanwendungsaufrufe* bei *Schritte für Fachanwendungsaufwurf (SOAP-Call)* als Expression angegeben werden.

```
FSCBAI@1.1001:InitBA(::dept);
```

Alternativ zur Verwendung der Aktion `FSCBAI@1.1001:InitBA` kann die gesamte Funktionalität des oben angeführten Aufrufs auch durch folgende Expression unter Verwendung von *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich*

(`FSCBAI@1.1001:CallSoapXmlEx`) abgebildet werden:

```
::params=coort.CreateDictionary();
::response=coort.CreateDictionary();
::params.SetEntryValue("objaddress",0,"Adresse")
::params.SetEntryValue("objname",0,"Name");
::params.SetEntryValue("deptref",0,::deptref);
```

```
FSCBAI@1.1001:CallSoapXmlEx(::dept, "Init", ::params, &::response);
```

```
FSCBAI@1.1001:rs.FSCBAI@1.1001:foreignkey = ::response.foreignkey;
FSCBAI@1.1001:rs.FSCBAI@1.1001:rsdept = ::deptref
```

### 12.1.2 Aufruf einer SOAP-Aktion mit XML-Parametern

Die Aktion *Aufruf einer SOAP-Aktion mit XML-Parametern* (FSCOWS@1.1001:CallSoapXml) ist eine Basisaktion für den Aufruf eines Webservice.

Parametername	IN/OUT	Typ	Erklärung
url	In	STRING	Endpunkt-URL des SOAP-Aufrufs
httpheaders	In	DICTIONARY	Zu sendende HTTP-Headers, wie z. B. „SOAPAction“. Die Header „Content-Type“ und „Content-Length“ werden automatisch berechnet und müssen daher nicht angegeben werden.
soapheaders	In	CONTENTLIST	Optionale Liste von SOAP-Headern. Ein SOAP-Header ist ein XML-Dokument-Fragment, das in die resultierende SOAP-Nachricht eingefügt wird.
xmlin	In		XML-Dokument-Fragment, das im Envelope/Body-Element versendet werden soll.
xmlout	Out		XML-Dokument-Fragment unterhalb des Envelope/Body-Elementes, das von der SOAP-Aktion zurückgegeben wurde.
parentwindow	In	INTEGER	Optionales Window-Handle des Vater-Fensters für die Ausgaben von Dialog-Fenstern für die Authentifizierung.
user	In	STRING	Optionaler Benutzername für die Authentifizierung am Webserver
password	In	STRING	Passwort des Benutzers
proxyuser	In	STRING	Optionaler Benutzername für die Authentifizierung am Proxyserver
proxypassword	In	STRING	Passwort des Proxybenutzers

clientcert	In	OBJECT	Optionales Client-Zertifikat, falls die Authentifizierung über ein Zertifikat erfolgen soll.
timeoutresolve	In	INTEGER	Zeitlimit in Sekunden für das Auflösen von DNS-Namen (Standardwert: kein Zeitlimit)
timeoutconnect	In	INTEGER	Zeitlimit in Sekunden für das Verbinden zum Webserver (Standardwert: 60 Sekunden)
timeoutsend	In	INTEGER	Zeitlimit in Sekunden für das Versenden. Es wird die Zeitdifferenz zwischen den Paketen gemessen. (Standardwert: 30 Sekunden)
timeoutreceive	In	INTEGER	Zeitlimit in Sekunden für das Empfangen. Es wird die Zeitdifferenz zwischen den Paketen gemessen. (Standardwert: 30 Sekunden)

### 12.1.3 Senden einer einfachen SOAP-Nachricht

Die Aktion *Sende eine einfache SOAP-Nachricht* (`FSCOWS@1.1001:CallSoapXmlEx`) bietet grundsätzlich dieselbe Funktionalität wie `FSCOWS@1.1001:CallSoapXml`. Beim Aufruf der SOAP-Aktion wird das WSDL-Dokument des aufgerufenen Webservice ausgewertet.

Parametername	IN/OUT	Typ	Erklärung
wSDLconnector	In	OBJECT	Ein Objekt der Objektklasse <i>HTTP-Anschluss</i> ( <code>FSCOWS@1.1001:HttpConnector</code> )
soapconnector	In	OBJECT	Optionaler Parameter, in dem ein Objekt der Objektklasse <i>HTTP-Anschluss</i> ( <code>FSCOWS@1.1001:HttpConnector</code> ) angegeben wird. Falls in diesem Parameter nichts angegeben wird, wird derselbe Anschluss wie im ersten Parameter verwendet.
operation	In	STRING	Name der auszuführenden Operation entsprechend dem WSDL-Dokument
inparams	In		Optionaler Eingangsparameter

		Folgende Typen sind möglich: OBJECT, STRING, CONTENT, Content und DICTIONARY
outparams	Out	Optionaler Ausgangsparameter Folgende Typen sind möglich: OBJECT, STRING, CONTENT, Content und DICTIONARY

Wird kein Ausgangsparameter angegeben, so wird der gleiche Typ wie im Eingangsparameter als Ausgangsparameter zurückgeliefert. Um einen bestimmten Ausgangstyp zu erhalten, muss ein leerer Wert des betreffenden Typs im Parameter `outparams` angegeben werden.

Beschreibung der Typen des Eingangs- und Ausgangsparameters:

- OBJECT  
Ein Objekt wird mithilfe der COOXML-Komponente auf ein XML-Dokument serialisiert. Das Mapping wird aus der XML-Default-Konfiguration aus der Eigenschaft *Zuordnung für Objektausgabe* gelesen.
- STRING  
XML-Dokument als Zeichenkette
- CONTENT  
XML-Dokument als Inhalt
- Content  
XML-Dokument in einem Inhaltsaggregat
- DICTIONARY  
Einträge im Dictionary werden mithilfe des XML-Schemas im WSDL-Dokument auf ein XML-Dokument abgebildet.

#### 12.1.4 Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich

Die Aktion *Aufrufen einer SOAP-Aktion mit verschiedenen Parametern über Fachanwendungsbereich* (`FSCBAI@1.1001:CallSoapXmlEx`) ruft die Aktion `FSCOWS@1.1001:CallSoapXmlEx` auf und bietet grundsätzlich dieselbe Funktionalität wie diese Aktion. Der Unterschied ist, dass statt den Parametern `wsdlconnector` und `soapconnector` nur der *Fachanwendungsbereich* angegeben werden muss. Die weiteren Informationen werden aus der *Konfiguration für Fachanwendungen* aus der Eigenschaft *Basis-URLs* ausgelesen.

Parametername	IN/OUT	Typ	Erklärung
dept	In	OBJECT	Ein Objekt der Objektklasse <i>Fachanwendungsbereichsdefinition</i>
operation	In	STRING	Name der auszuführenden Operation entsprechend dem WSDL-Dokument

inparams	In	Optionaler Eingangsparameter Folgende Typen sind möglich: OBJECT, STRING, CONTENT, Content und DICTIONARY
outparams	Out	Optionaler Ausgangsparameter Folgende Typen sind möglich: OBJECT, STRING, CONTENT, Content und DICTIONARY

### 12.1.5 Laden des Objekts aus einem CONTENT mit XML-Daten

Die Aktion *Laden des Objekts aus einem CONTENT mit XML-Daten*

(FSCBAI@1.1001:MapContentToObject) kann für das Mapping von einem Objekt auf ein XML-Dokument verwendet werden.

Parametername	IN/OUT	Typ	Erklärung
dept	In	OBJECT	Ein Objekt der Objektklasse <i>Fachanwendungsbereichsdefinition</i>
ctx	In	OBJECT	Kontext, um die passende Konfigurationszeile in der <i>Konfiguration für Fachanwendungen</i> (FSCBAI@1.1001:ConfigurationClass) zu finden
content	In	CONTENT	XML-Inhalt als CONTENT

Diese Aktion wird auf das Objekt angewendet, das auf den XML-Inhalt abgebildet wird. Das anzuwendende XML-Mapping wird aus der *Konfiguration für Fachanwendungen* anhand der Objektklasse und dem Auslöser ausgelesen.

### 12.1.6 Ausgeben des Objekts in einen CONTENT mit XML-Daten

Die Aktion *Ausgeben des Objekts in einen CONTENT mit XML-Daten*

(FSCBAI@1.1001:MapObjectToContent) kann für das Mapping eines XML-Dokuments auf ein Objekt verwendet werden.

Parametername	IN/OUT	Typ	Erklärung
dept	In	OBJECT	Ein Objekt der Objektklasse <i>Fachanwendungsbereichsdefinition</i>

ctx	In	OBJECT	Kontext um die passende Konfigurationszeile in der <i>Konfiguration für Fachanwendungen</i> zu finden
content	Out	CONTENT	XML-Inhalt als CONTENT

Diese Aktion wird auf das Objekt angewendet, auf das der XML-Inhalt abgebildet wird. Das anzuwendende XML-Mapping wird aus der *Konfiguration für Fachanwendungen* anhand der Objektklasse und dem Auslöser ausgelesen.

### 12.1.7 Kombiniertes SOAP-Client-/GUI-Aufruf

In der *Konfiguration für Fachanwendungen* wird ein SOAP-Client-Aufruf und ein anschließender GUI-Aufruf in der Eigenschaft *Fachanwendungsaufrufe* definiert und konfiguriert.

Der Aufruf erfolgt über die Aktion *Fachanwendung über Programmiername starten* (FSCBAI@1.1001:StartBusinessApp) mit folgendem Parameter:

Parametername	IN/OUT	Typ	Erklärung
prognose	In	STRING	Programmiername

Die Aktion entscheidet anhand der Objektklasse des Objekts, auf dem sie ausgeführt wird, welcher Konfigurationseintrag in der *Konfiguration für Fachanwendungen* (in der Eigenschaft *Fachanwendungsaufrufe*) ausgewertet und ausgeführt wird (best-matching Objektklasse). Dadurch können ein SOAP-Aufruf und ein URL-Aufruf an die Fachanwendung durchgeführt werden.

#### Beispiel:

Die Anwendung wird der Variable `app` zugewiesen. Dazu muss der Programmiername des Aufrufs einer Variablen als Zeichenkette zugewiesen werden. Aufgrund der späteren Übergabe des Parameters „by Name“ ist die strikte Einhaltung des Bezeichners `prognose` notwendig. Der Inhalt der Zeichenkette („editcase“) muss mit dem Programmiernamen der Eigenschaft *Fachanwendungsaufrufe* übereinstimmen.

Der Anweisungsschritt `FSCVAPP@1.1001:CallAppStep` muss auf das Objekt, bei dem sich der Branch befindet (`sys_branchvalue`), ausgeführt werden. Dabei muss die Variable mit dem Anwendungsobjekt übergeben werden; der Programmiername wird „by Name“ übergeben.

#### Beispiel:

Aufruf der Fachanwendung über ein Skript (z.B. als Arbeitsschritt):

```
//LANGUAGE="JScript"
var inboundobj = coort.GetObject("COO.1.1019.2.1002890");
var progname = "editoffense"
var meth=inboundobj.GetMethod(coortx, "FSCBAI@1.1001:StartBusinessApp");
```

## 13 Funktionsteilbereich „Konfiguration“

Für die Konfiguration der Fachanwendungsintegration ist folgendes Konfigurationsobjekt relevant:

- *Konfiguration für Fachanwendungen* (FSCBAI@1.1001:ConfigurationClass)  
In diesem Konfigurationsobjekt werden alle direkt zur Fachanwendungsintegration gehörenden Einstellungen vorgenommen (z. B. Definition von Kontextmenüs).

### 13.1 Konfiguration für Fachanwendungen

Innerhalb der *Konfiguration für Fachanwendungen* (FSCBAI@1.1001:ConfigurationClass) werden Werte über Fabasoft DUCX Expressions ermittelt. Zur Vereinheitlichung steht in allen Fabasoft DUCX Expressions dieser Konfiguration im globalen Scope ein Dictionary zur Verfügung, in dem – abhängig von der aktuellen Situation – die folgenden Werte vorhanden sind:

Dictionary	Beschreibung
::data	<i>Fachanwendungsdaten-Aggregat</i> des aktuellen Objekts
::dept	<i>Fachanwendungsbereich</i>
::deptref	Vollständige <i>Referenz</i> des <i>Fachanwendungsbereichs</i> als Zeichenkette

Im *lokalen Scope* (`this`) steht das jeweilige aktuelle Objekt zur Verfügung.

Im Folgenden werden die in der Konfiguration für Fachanwendungen zur Verfügung stehenden Eigenschaften näher beschrieben.

#### 13.1.1 URL für lokales WebDAV

In dieser Eigenschaft wird der URL, unter dem die WebDAV-Schnittstelle der Fabasoft Produktinstallation Installation erreichbar ist, angegeben. Diese Einstellung ist für das Erzeugen und für die Anzeige von *Fachdokumenten* erforderlich. Das Menü zum Anzeigen von *Fachdokumenten* ruft die Aktion *Transformierten XML-Inhalt ermitteln*

(FSCBAI@1.1001:GetPrimaryContents) über WebDAV auf.

##### Beispiel:

So sieht der URL für lokales WebDAV von einem Fabasoft Folio Webservice aus:

```
http://<webserver>/<vdir>/fscdav/CALL
```

#### 13.1.2 Basis-URLs

In dieser Eigenschaft können pro Fachanwendung die Endpunkte für den GUI-Aufruf sowie den SOAP-Aufruf angegeben werden. Zur Definition der Endpunkte der SOAP-Aufrufe werden *HTTP-Anschluss-Konfigurationen* (FSCOWS@1.1001:HttpConnector) verwendet. Für die einzelnen *Fachanwendungsbereiche* können unterschiedliche HTTP-Anschlüsse angegeben werden. Dies ist sinnvoll, wenn die Fachanwendung über verschiedene Webservices verfügt.

## Fachanwendungsbereich

In dieser Eigenschaft wird ein *Fachanwendungsbereich* eingetragen. Dieser *Fachanwendungsbereich* identifiziert eine Fachanwendung oder einen Teil davon.

## Softwarekomponente

Damit die getroffenen Änderungen bei einem Update nicht verloren gehen, muss in dieser Eigenschaft eine *Softwarekomponente* eingetragen werden. Bei dieser *Softwarekomponente* muss `FSCBAI@1.1001` als *vorausgesetzte Softwarekomponente* eingetragen sein.

## Basis-URL für GUI

Der in dieser Eigenschaft eingetragene Basis-URL wird mit den Parametern vervollständigt, die in der Eigenschaft *Fachanwendungsaufrufe* in der *Konfiguration für Fachanwendungen* als URL-Parameter eingetragen sind.

## HTTP-Anschluss für SOAP-Schnittstelle

In der Objektzeigereigenschaft *HTTP-Anschluss für SOAP-Schnittstelle* (`FSCBAI@1.1001:baseurlsoapif`) wird festgelegt, über welchen URL ein SOAP-Call von der Fabasoft Folio Domäne zur Fachanwendung abgesetzt werden kann.

Hier werden Komponentenobjekte des Typs *HTTP-Anschluss-Konfiguration* (`FSCOWS@1.1001:HttpConnector`) eingetragen.

- *HTTP Anschluss*  
Die zusammengesetzte Eigenschaft *HTTP-Anschluss* (`FSCOWS@1.1001:httpconsingel`) beschreibt die Verbindungseigenschaft zu einer Fachanwendung. Neben dem URL können Benutzer bzw. Proxy-Benutzer und Zeitlimits gesetzt werden.
- *URL*  
In der Zeichenketteneigenschaft *URL* (`FSCOWS@1.1001:httpconnurl`) wird die Adresse des Ziel-Webservice der Fachanwendung definiert.
- *Benutzer*  
In der Zeichenketteneigenschaft *Benutzer* (`FSCOWS@1.1001:httpconnuser`) kann ein Benutzer definiert werden, der die entsprechenden Berechtigungen beim Ziel-Webservice besitzt.
- *Passwort*  
In der Zeichenketteneigenschaft *Passwort* (`FSCOWS@1.1001:httpconnpassword`) wird das Passwort des Benutzers angegeben.
- *Proxy-Benutzer*  
In der Zeichenketteneigenschaft *Proxy-Benutzer* (`FSCOWS@1.1001:httpconnproxyuser`) kann ein Proxy-Benutzer definiert werden, der die entsprechenden Berechtigungen beim Ziel-Webservice besitzt. Hier muss die Authentifizierung über einen Proxy-Server stattfinden.
- *Proxy-Benutzer-Passwort*  
In der Zeichenketteneigenschaft *Proxy-Benutzer-Passwort* (`FSCOWS@1.1001:httpconnproxypassword`) wird das Passwort des Proxy-Benutzers angegeben.

- *Client-Zertifikat*  
In der Objekteigenschaft *Client-Zertifikat* (FSCOWS@1.1001:httpconncertificate) kann ein Client-Zertifikat hinterlegt werden, sofern das Ziel-Webservice Client-Zertifikate unterstützt.
- *Zeitlimit für Auslösen (in Sek)*  
In der Zahleneigenschaft *Zeitlimit für Auslösen (in Sek)* (FSCOWS@1.1001:httpconntimeoutresolve) wird ein Zeitlimit definiert, das für das Auslösen berücksichtigt wird.
- *Zeitlimit für Verbinden (in Sek)*  
In der Zahleneigenschaft *Zeitlimit für Verbinden (in Sek)* (FSCOWS@1.1001:httpconntimeoutconnect) wird ein Zeitlimit definiert, das für den Verbindungsaufbau berücksichtigt wird.
- *Zeitlimit für Senden (in Sek)*  
In der Zahleneigenschaft *Zeitlimit für Senden (in Sek)* (FSCOWS@1.1001:httpconntimeoutsend) wird ein Zeitlimit definiert, das für das Senden berücksichtigt wird.
- *Zeitlimit für Empfangen (in Sek)*  
In der Zahleneigenschaft *Zeitlimit für Empfangen (in Sek)* (FSCOWS@1.1001:httpconntimeoutreceive) wird ein Zeitlimit definiert, dass für das Empfangen einer Antwort berücksichtigt wird.
- *Softwarekomponente*  
In der Objekteigenschaft *Softwarekomponente* (FSCOWS@1.1001:httpconnswo) wird eine lokale Softwarekomponente referenziert, damit die im *HTTP-Anschluss*-Objekt definierten Eintragungen auch nach einem Update der Fabasoft Folio Domäne weiterhin zur Verfügung stehen.

### HTTP-Anschluss für WSDL

Diese Eigenschaft legt fest, unter welchem URL die von der Fachanwendung zur Verfügung gestellte Webservice-Definition (WSDL) bereitgestellt wird. Diese Information wird von der Aktion *Sende eine einfache SOAP-Nachricht* (FSCBAI@1.1001:CallSoapXmlEx) ausgewertet. Dabei wird anhand des *Fachanwendungsbereichs* der HTTP-Anschluss für WSDL ermittelt. Die Informationen aus dem WSDL-Dokument werden u.a. für die Verwendung der richtigen Namespaces im SOAP-Request benötigt und entsprechend an FSCOWS@1.1001:CallSoapXmlEx weitergereicht. Diese Aktion ruft anschließend die angegebene SOAP-Aktion auf.

### 13.1.3 Fachanwendungsaufufe

In dieser Eigenschaft werden die angezeigten Einträge im dynamischen Kontextmenü *Fachanwendungen* (FSCBAI@1.1001:MenuBusinessApps) (durch die Aktion FSCBAI@1.1001:MenuBusinessApp) sowie die Aufrufe, die mit der *Aktion Fachanwendung über Programmiername starten* (FSCBAI@1.1001:StartBusinessApp) über den Programmiernamen gestartet werden, konfiguriert.

Bei einigen Objektklassen ist der Menübefehl *Fachanwendungen* bereits eingetragen. Falls bei einer Objektklasse dieser Kontextmenübefehl noch nicht vorhanden ist und hinzugefügt werden soll, muss in das Menü, das bei der entsprechenden Objektklasse in der Eigenschaft

*Kontextmenü für Container in Fabasoft Folio* (COODESK@1.1:classctxmenu) hinterlegt ist, das Menü *Fachanwendungen* (FSCGOVB@1.1001:MenuBusinessApps) bzw. das Menü *Fachdokumente* (FSCGOVB@1.1001:MenuDocuments) eingetragen werden.

Die im Folgenden beschriebenen Eigenschaften stehen in der zusammengesetzten Eigenschaft *Fachanwendungsaufrufe* (FSCBAI@1.1001:menuconfig) der *Konfiguration für Fachanwendungen* zur Verfügung.

### Programmiername

Für unterschiedliche Objektklassen kann derselbe Programmiername mehrfach verwendet werden. Die Aktion FSCBAI@1.1001:StartBusinessApp entscheidet anhand der Objektklasse des Objekts, auf dem sie ausgeführt wird, welcher Aufruf gültig ist (best-matching Objektklasse). Ein Branch einer vApp kann z. B. allgemeingültig einen Fachanwendungsaufruf durchführen. Dieser Aufruf wird weiter für bestimmte Objektklassen genauer spezifiziert.

### Beispiel:

Es soll ein *Fachanwendungsaufruf* aus einem JavaScript durchgeführt werden. Die Zeile in der *Konfiguration für Fachanwendungen* hat als Programmiernamen „editoffense“ eingetragen.

```
//LANGUAGE="JScript"  
var inboundobj = coort.GetObject("COO.1.1019.2.1002890");  
var progname = "editoffense"  
var meth=inboundobj.GetMethod(coortx, "FSCBAI@1.1001:StartBusinessApp");  
meth.SetParameterValue(1,"COOSYSTEM@1.1:STRING", 0, progname);  
inboundobj.CallMethod(coortx, meth);
```

### Objektklasse

Die Angabe einer Objektklasse ist für die Bildung des Kontextmenüs relevant – nur wenn das Objekt der entsprechenden Objektklasse entspricht, wird der Menübefehl weiter behandelt. Weiters ist sie ein Schlüssel, der einen Eintrag innerhalb des Aggregats eindeutig macht.

### Fachanwendungsbereich

Für die Darstellung im Menü muss der im *Fachanwendungsaufruf* angegebene *Fachanwendungsbereich* verfügbar sein – d.h. der *Fachanwendungsbereich* muss entweder in der *aktuellen Domäne*, in der *Gruppe des Benutzers* oder in der *Arbeitsumgebung* eingetragen sein.

### Softwarekomponente

Die *Softwarekomponente* muss angegeben werden, damit die Einstellungen bei einem Update der Fabasoft Folio Domäne nicht verloren gehen.

### Mehrsprachiger Name

Dieser Eintrag legt die Bezeichnung fest, die im Kontextmenü dargestellt wird. Wenn der Menüeintrag nicht im Kontextmenü angezeigt werden soll, muss der *Ausdruck für versteckten Menüeintrag (nicht sichtbar wenn...)* (FSCBAI@1.1001:invisibleexpr) im Aggregat *Erscheinungsbild* (FSCBAI@1.1001:appearance) beachtet werden.

**Hinweis:** Der Bezeichnung des Menüeintrags kann eine Zugriffstaste zugewiesen werden, indem vor ein Zeichen ein &-Zeichen gestellt wird.

### Schritte für Fachanwendungsaufruf (SOAP-Call)

Diese Eigenschaft beinhaltet eine Fabasoft DUCX Expression, die einen SOAP-Aufruf an die Fachanwendung durchführt.

#### **Beispiel:**

```
FSCBAI@1.1001:InitBA(::dept);
```

Die weitere Vorbereitung der Daten, der Aufruf und die Nachbearbeitung der Daten erfolgt durch die Implementierung der Aktion *Standardinitialisierung eines Objekts über SOAP* (FSCBAI@1.1001:InitBA).

Der Aufruf kann alternativ dazu unter Verwendung der Aktion FSCBAI@1.1001:CallSoapXmlEx generisch abgebildet werden:

#### **Beispiel:**

```
// Vorbereiten der Eingangsdaten
::params = coort.CreateDictionary;
::params.objaddress = this.objaddress;
::params.objname = this.objname;
::params.deptref = ::deptref;
// Aufruf der SOAP-Aktion
::response= this.FSCBAI@1.1001:CallSoapXmlEx(::dept, "Init", ::params)[4];
// Setzen des Fremdschlüssels
this.FSCBAI@1.1001:rs.FSCBAI@1.1001:foreignkey = ::response.foreignkey;
this.FSCBAI@1.1001:rs.FSCBAI@1.1001:rsdept = ::deptref
```

Anfangs werden die Eingangsdaten (Eingangsparameter in Form von XML) erstellt und einem Dictionary zugewiesen. Dieses Dictionary wird im Laufe des Aufrufs in XML konvertiert und der Aktion als Request-Parameter mitgegeben.

Anschließend wird die SOAP-Aktion mit der Operation `Init` aufgerufen. Die Namen von Operationen sind in der WSDL des Ziel-Webservice definiert.

Danach wird ein *Fachanwendungs-Datensatz* durch das Setzen von `FSCBAI@1.1001:rs` initialisiert.

**Anmerkungen:** Der Operator für den lokalen Scope (`this`) ist optional. Das Aggregat `FSCBAI@1.1001:rs` schreibt über eine Set-Aktion die Daten in das Aggregat `FSCBAI@1.1001:data`. Dabei wird ein neuer Datensatz erzeugt und der Fachanwendungsbereich anhand der Referenz ermittelt (`::deptref` beinhaltet die vollständige *Referenz des Fachanwendungsbereichs* als Zeichenkette).

#### **Ausdruck für GUI-Aktion (URL)**

Dieser Wert wird im `ax`-Parameter des URL an die Fachanwendung übergeben. Soll Fabasoft Folio als Fachanwendung verwendet werden, muss hier die Objektadresse der vApp als Zeichenkette (d.h. innerhalb von Hochkomma) eingetragen werden.

#### **Ausdruck für optionale GUI-Parameter**

Diese Eigenschaft beinhaltet weitere Parameter, die im URL übergeben werden. Bei .NET-Fachanwendungen muss diese Eigenschaft nicht angegeben werden. Eine mögliche Anwendung wäre die Definition eines eigenen Web-Dispatchers bei einer Fabasoft Folio Fachanwendung.

#### **Ausdruck für Return-URL (Post-GUI)**

Dieser Ausdruck gibt an, zu welchem URL die Fachanwendung nach der Darstellung zurückkehren soll.

#### **Beispiel:**

```
javascript>window.close()
```

Das Fenster wird nach dem Abarbeiten aller Operationen geschlossen.

#### **Erscheinungsbild**

In dieser Eigenschaft wird festgelegt, ob und wie ein Menüeintrag dargestellt werden soll. Für jeden *Fachanwendungsaufwurf* muss ein Erscheinungsbild festgelegt sein.

- *Separator vorausgehend*  
Die Eigenschaft *Separator vorausgehend* gibt an, ob vor dem Menüeintrag eine Trennlinie angezeigt werden soll.
- *Ausdruck für versteckten Menüeintrag (nicht sichtbar wenn...)*  
In dieser Eigenschaft wird festgelegt, unter welchen Umständen der Menüeintrag nicht angezeigt wird. Ist die Auswertung dieser Expression `TRUE`, dann wird der Menüeintrag nicht angezeigt, bei `FALSE` hingegen wird er angezeigt.  
**Anmerkung:** Ein Menüeintrag wird dann dargestellt, wenn der *Fachanwendungsbereich* für den einzelnen Benutzer verfügbar ist (d.h. der *Fachanwendungsbereich* muss entweder in der *aktuellen Domäne*, in der *Gruppe* des Benutzers oder in der *Arbeitsumgebung* eingetragen sein).
- *Ausdruck für deaktivierten Menüeintrag (nicht verfügbar wenn...)*  
Dieser Ausdruck gibt an, unter welchen Umständen der Menüeintrag nicht verfügbar dargestellt wird. Ist die Auswertung dieser Expression `TRUE`, dann wird der Menüeintrag inaktiv dargestellt, bei `FALSE` hingegen wird er aktiv angezeigt.
- *Pfad zur Fachanwendungsbereichsliste*  
Die Eigenschaft *Pfad zur Fachanwendungsbereichsliste* beinhaltet eine Liste von Eigenschaftsobjekten, die als Eigenschaftspfad interpretiert werden – ausgehend vom aktuellen Objekt, auf das der Menübefehl ausgeführt wird. Dieser Pfad führt vom aktuellen Objekt zu einer Objektliste, die im *Fachanwendungsbereich* enthalten ist. Ist ein Eigenschaftspfad eingetragen, wird der Menübefehl nur angezeigt, wenn der beim Menüeintrag angegebene *Fachanwendungsbereich* auch in der über den Pfad angegebenen Objektliste enthalten ist.

### **13.1.4 Menü für Fachdokumente**

*Fachdokumente* im HTML-Format werden über eine XSL-Transformation aus den *Fachanwendungsdaten* erstellt. Das HTML-Dokument wird über WebDAV im Webbrowser angezeigt – es wird kein neues Objekt in Fabasoft Folio erzeugt. In der *Konfiguration für Fachanwendungen* muss dazu ein *URL für WebDAV* angegeben werden.

Bei einigen Objektklassen ist der Menübefehl *Fachdokumente* bereits eingetragen. Falls bei einer Objektklasse dieser Kontextmenübefehl noch nicht vorhanden ist und hinzugefügt werden soll, muss in das Menü, das bei der entsprechenden Objektklasse in *Kontextmenü für Container in Fabasoft Folio* hinterlegt ist, das Menü *Fachdokumente* (`FSCBAI@1.1001:MenuDocuments`) eingetragen werden.

Die im Folgenden beschriebenen Eigenschaften stehen in der zusammengesetzten Eigenschaft *Menü für Fachdokumente* zur Verfügung.

### **Objektklasse**

Die Angabe einer Objektklasse ist für die Bildung des Kontextmenüs relevant – nur wenn das Objekt der angegebenen Objektklasse entspricht, wird der Menübefehl weiterbehandelt.

### **Fachanwendungsbereich**

Für die Darstellung im Menü muss der im *Fachanwendungsaufruf* angegebene *Fachanwendungsbereich* verfügbar sein – d.h. der *Fachanwendungsbereich* muss entweder in der *aktuellen Domäne*, in der *Gruppe* des Benutzers oder in der *Arbeitsumgebung* eingetragen sein.

### **Softwarekomponente**

Die *Softwarekomponente* muss angegeben werden, damit die Einstellungen bei einem Update der Fabasoft Folio Domäne nicht verloren gehen.

### **Mehrsprachiger Name**

Dieser Eintrag legt die Bezeichnung fest, die im Kontextmenü dargestellt wird. Wenn der Menüeintrag nicht im Kontextmenü angezeigt werden soll, muss der *Ausdruck für versteckten Menüeintrag* (*nicht sichtbar wenn...*) im Aggregat *Erscheinungsbild* definiert werden.

**Hinweis:** Der Bezeichnung des Menüeintrags kann eine Zugriffstaste zugewiesen werden, indem vor ein Zeichen ein &-Zeichen gestellt wird.

### **Kontext**

In dieser Eigenschaft wird ein beliebiges *Komponentenobjekt* angegeben, das beim Aufruf des Menübefehls an die Aktion *Transformierten XML-Inhalt ermitteln* (FSCBAI@1.1001:GetPrimaryContents) übergeben wird. Diese Aktion ermittelt anhand der angegebenen Objektklasse und dem Kontext eine geeignete XSL-Transformation in der Eigenschaft *Transformation für Fachdokumente*.

### **Erscheinungsbild**

In dieser Eigenschaft wird festgelegt, ob und wie ein Menübefehl dargestellt werden soll. (Nähere Informationen dazu siehe Kapitel 13.1.3 „Fachanwendungsaufrufe“.)

## **13.1.5 Transformation für Fachdokumente**

Diese Eigenschaft ist eng mit der Eigenschaft *Menü für Fachdokumente* (FSCBAI@1.1001:documentsmenu) in der *Konfiguration für Fachanwendungen* verbunden. Hier wird die eigentliche Transformation festgelegt.

### **Objektklasse**

Die betreffende Objektklasse für die Transformation wird in dieser Eigenschaft angegeben. Die Transformation kann also nur bei Objekten der angegebenen Objektklasse verwendet werden.

### **Fachanwendungsbereich**

Für die Darstellung im Menü muss der im Fachanwendungsaufruf angegebene *Fachanwendungsbereich* verfügbar sein – d.h. der *Fachanwendungsbereich* muss entweder in

der *aktuellen Domäne*, in der *Gruppe* des Benutzers oder in der *Arbeitsumgebung* eingetragen sein.

### Softwarekomponente

Die *Softwarekomponente* muss angegeben werden, damit die Einstellungen bei einem Update der Fabasoft Folio Domäne nicht verloren gehen.

### Kontext

In dieser Eigenschaft wird ein *Komponentenobjekt* angegeben, das an die Aktion FSCBAI@1.1001:GetPrimaryContents übergeben wird. Nähere Informationen dazu siehe Kapitel 13.1.4 „Menü für Fachdokumente“.

### Transformationsobjekt (XSLT)

In dieser Eigenschaft wird ein Inhaltsobjekt angegeben, das eine XSL-Transformation enthält. Diese Transformation wird verwendet, um aus den *Fachanwendungsdaten* ein HTML-Dokument zu generieren.

### Transformation (XSLT)

Wird kein bestehendes Transformationsobjekt angegeben oder erfolgt die Transformation abhängig von der eingestellten Sprache, kann in dieser Eigenschaft unter Angabe einer Sprache eine XSL-Transformation als Inhalt hinterlegt werden.

### Beispiel:

So könnte eine XSL-Transformation aussehen, um ein *Fachdokument* zu erzeugen.

```
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'
xmlns:ns1="urn:schemas-fabasoft-com:bai:fachanwendung-straften">
  <xsl:output method='html' encoding='ISO-8859-1' />
  <xsl:template match='/'>
    <html>
      <body>
        <p style="font-weight:bold">Strafakt</p>
        <table width="100%" border="1">
          <tr>
            <td><p>Betreff:</p></td> <td><p><xsl:value-of
              select="//ns1:data/ns1:casefile/ns1:subject"/></p></td>
          </tr>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

**Hinweis:** Beim Speichern der Datei muss das Format UTF-8 verwendet werden.

## 13.1.6 Weiterführen von permanenten Sperren

Diese Eigenschaft in der *Konfiguration für Fachanwendungen* wird nur bei Verwendung der SOAP-Aktion *Auschecken über SOAP* (FSCBAI@1.1001:SOAPCheckOut) berücksichtigt.

Werden Objekte mit Objektliste gesperrt, können auch enthaltene Objekte mitgesperrt werden.

### Objektklasse

Die betreffende Objektklasse für das Weiterführen von Sperren wird in dieser Eigenschaft festgelegt.

#### **Auslöser**

In dieser Eigenschaft wird ein *Komponentenobjekt* angegeben, in dessen Kontext eine Sperre auf Objekte weitergeführt werden soll. Derzeit werden die in dieser Eigenschaft vorgenommenen Einstellungen von der Aktion *Auschecken über SOAP* (FSCBAI@1.1001:SOAPCheckOut) verwendet. Die Aktion wird dabei selbst als Kontext verwendet.

#### **Softwarekomponente**

Die *Softwarekomponente* muss angegeben werden, damit die Einstellungen bei einem Update der Fabasoft Folio Domäne nicht verloren gehen.

#### **Ausdruck für Objekte, auf die die Sperre angewendet werden soll**

In dieser Eigenschaft wird eine Fabasoft DUCX Expression angegeben, die eine Liste von Objekten (keine Eigenschaftsdefinition als Objektliste) zurückgibt.

#### **Beispiel:**

Es soll eine Objektliste der Objekte zurückgegeben werden, die gesperrt werden sollen.

```
::items=COOSYSTEM@1.1:GetModifyPropagation()[1];  
::items
```

**Anmerkung:** Bei einer Objektklasse kann angegeben werden, welche Objektlisten bei Weiterführungen beachtet werden sollen (*Weiterführende Änderungen von Eigenschaften*). Die in den angegebenen Objektlisten enthaltenen Objekte können mit der Aktion *Lesen der Liste der von weiterführenden Änderungen betroffenen Objekte* (COOSYSTEM@1.1:GetModifyPropagation) ermittelt werden.

### **13.1.7 Schematazuordnungen**

Mit der Aktion *Mapping aus Fachanwendungs-Konfiguration ermitteln* (FSCBAI@1.1001:GetBAIMapping) wird ein XML-Mapping ermittelt. Die hier eingetragenen Mappings werden von SOAP-Aktionen der Softwarekomponente *Integration for Business Application* (FSCBAI@1.1001) verwendet.

#### **Objektklasse**

Die betreffende Objektklasse für die Schemazuordnungen wird in dieser Eigenschaft festgelegt.

#### **Auslöser**

In dieser Eigenschaft wird eine SOAP-Aktion angegeben, die die in der Eigenschaft *Zuordnung* angegebene Zuordnung nutzt.

**Hinweis:** Die beiden SOAP-Aktionen *Einchecken über SOAP* (FSCBAI@1.1001:SOAPCheckIn) und *Auschecken über SOAP* (FSCBAI@1.1001:SOAPCheckOut) und die beiden Aktionen *Ausgeben des Objekts in einen CONTENT mit XML-Daten* (FSCBAI@1.1001:MapObjectToContent) und *Laden des Objekts aus einem CONTENT mit XML-Daten* (FSCBAI@1.1001:MapContentToObject) verwenden diese Konfigurationseinträge um das zu verwendende XML-Schema und die zu verwendende XML-Zuordnung zu ermitteln.

### **Fachanwendungsbereich**

Die Verwendung einer XML-Zuordnung kann abhängig vom *Fachanwendungsbereich* erfolgen, der in dieser Eigenschaft angegeben wird. Die jeweilige Implementierung kann diesen Parameter berücksichtigen und eine vom *Fachanwendungsbereich* abhängige XML-Zuordnung wählen.

### **Softwarekomponente**

Die *Softwarekomponente* muss angegeben werden, damit die Einstellungen bei einem Update der Fabasoft Folio Domäne nicht verloren gehen.

### **Zuordnung**

In dieser Eigenschaft wird eine *Abbildung von XML-Elementen auf Objekteigenschaften* (XML-Mapping) angegeben, die im konkreten Fall verwendet werden soll. Die tatsächliche Verwendung (z. B. auf welchen Knoten der XML-Datei das Mapping angewendet wird) ist abhängig von der jeweiligen Implementierung der Aktion.

## **14 Glossar**

### **Fachanwendung**

Auf eine im Allgemeinen eher eng gefasste Aufgabenstellung spezialisierte Software. Den Gegensatz dazu bildet Software mit Querschnittsaufgaben wie Dokumentenmanagement- und Workflow-Funktionen.

### **Fachanwendungsdomäne**

Domäne der Fachanwendung, die mindestens einen *Fachanwendungsbereich* enthält.

### **HTML (Hyper Text Mark-up Language)**

HTML ist eine Sprache zur Auszeichnung von Hypertext im **World Wide Web (WWW)** und dem logischen Aufbau eines Dokuments, wie zum Beispiel Überschriften, Textabsätze, Listen und Tabellen.

### **Hyper Texttransport Protocol (HTTP) / Hypertext Transport Protocol Secure (HTTPS)**

HTTP ist ein Datenaustausch-Protokoll, mit dem ein Webbrowser auf einen Webserver zugreifen kann. HTTPS ist eine gesicherte HTTP-Verbindung, bei der die Daten über SSL/TLS verschlüsselt werden.

### **Konvertierung**

Überführung einer Datei eines bestimmten Formats wie zum Beispiel .doc, .xls, .ppt usw. in ein anderes Dateiformat wie zum Beispiel .pdf oder .html.

### **Metadaten**

Daten, die Bedeutung, Beschaffenheit, Struktur und den Aufbau anderer Daten beschreiben. Metadaten sind strukturierte oder teilweise unstrukturierte Information, die zum Beispiel den Prozess der Entstehung und Wirkung (Nutzung) von Unterlagen, ihren Aufbau sowie ihre Verwaltung und Verwahrung nachvollziehbar machen.

### **Objektklasse**

Jedes Objekt ist eine Instanz einer Objektklasse. Objektklassen sind immer von genau einer Objektklasse abgeleitet und erben von dort alle Eigenschaften und Methoden. Objektklassen sind in einem Baum angeordnet, der genau eine Wurzel *Objekt* (COOSYSTEM@1.1:Object) hat.

### **Stammdomäne**

Eine Fabasoft Folio Domäne mit einer Installation der Fabasoft Produktinstallation oder der Fabasoft eCRM-Suite, für die eine Fachanwendung entwickelt wird.

### **Transaktion**

Eine Abfolge von mehreren Operationen, die entweder vollständig oder gar nicht durchgeführt werden müssen. Passiert in einer Operation ein Fehler, müssen alle zuvor durchgeführten Änderungen wieder rückgängig gemacht werden. Werden alle Operationen ohne Fehler durchgeführt, werden alle Änderungen übernommen.

### **Use-Case**

Ein Use-Case ist die Beschreibung eines in Schritte untergliederten Anwendungsfalls eines Anwenders.

### **Webservice**

Ein auf Webstandards (in der Regel XML, HTTP) aufbauender Dienst zum Austausch von Informationen in einer verteilten und dezentralisierten Umgebung. Das Webservice kann von einem Client über das Internet mit einem **Uniform Resource Locator (URL)** angesprochen werden. Die Funktionalität des Webservices wird in einer definierten Schnittstelle - häufig mithilfe der Web Service Definition Language (WSDL) - beschrieben. Aufgrund der verwendeten Standards (XML und HTTP) sind Webservices unabhängig von einer bestimmten Programmiersprache und Systemplattform.

### **Extensible Markup Language (XML)**

Eine Sprache, mit der die Struktur von Dokumenten beschrieben wird (eine sogenannte Metasprache). XML wurde von der W3C als Datenformat festgelegt, um die einfache Implementierung in bestehende Software und die einfache Kommunikation zwischen mehreren Softwareprodukten zu gewährleisten.

### **XSL-Stylesheet (Extensible Stylesheet Language)**

Wurde entwickelt, um XML-Dateien konvertieren und ausgeben zu können. Das am häufigsten verwendete Zielformat ist HTML.