SoftENGINE

UPDATE

TECHNIK

Webservices

Ab Version WEBWARE 2.01 (Erprobungsphase 2016)

Philipp Müller, kaufmännischer Leiter der Hund-Katze-Goldfisch GmbH

Zukünftig wollen wir in verschiedenen Bereichen im Unternehmen auf die WEBWARE setzen und müssen natürlich unseren Webshop auch entsprechend anbinden.

In der BüroWARE haben wir es bislang über Standardschnittstellen & Datenbank-Zugriff direkt gelöst, in der WEBWARE helfen uns zukünftig Webservices. Diese werden von unserer IT selbst programmiert. Es geht also darum: Wie komme ich an Daten, die in der WEBWARE gespeichert sind und wie bekomme ich Daten hinein in die WEBWARE?

Wie kann beispielsweise unser Kunde online den aktuellsten Lagerbestand des gewünschten Artikels einsehen oder wie können Stammkunden im Kundenkonto Aktionspreise und Sonderpreise ersehen, die sich natürlich von den öffentlichen ShopPreisen unterscheiden?

Da die Preispflege komplett in der WEBWARE hinterlegt ist, kann unser Bestandskunde über einen Webservice-Call genau seinen Preis für diesen Artikel, an diesem Tag mit der gewünschten Menge einsehen. Die ganze Preisermittlung läuft mit Formeln und über Individualisierungen in der WEBWARE ab. Im Shop sieht der Kunde jedoch tatsächlich SEINEN aktuellen Preis. Was ebenfalls unschlagbar effizient ist, ist die Tatsache, dass die Online-Bestellungen direkt in unsere WEBWARE gelangen.

Und das alles ohne Synchronisation nach bestimmten Zeitintervallen. Da die SoftENGINE Webservices direkt integriert sind, haben sie erhebliche Geschwindigkeitsvorteile gegenüber Third Partie Lösungen.

Auch unseren Außendienst haben wir über WEBWARE Webservices angebunden. Mit einer externen APP auf den Tablets unserer Mitarbeiter kann mittels der Webservices DIREKT und absolut SICHER auf die zur Verfügung gestellten Daten unserer WEBWARE zugegriffen werden.

Das bedeutet in erster Hinsicht, dass wir InHouse Mitarbeiter im normalen Tagesgeschäft unsere Stammdaten bearbeiten und pflegen, spezielle Adresskonditionen anpassen u.v.m. und ohne dass eine Datenaufbereitung stattfinden muss, alle gespeicherten Daten in der WEBWARE, per Webservices sofort auch dem Außendienst zur Verfügung stehen.

Technische Details

In diesem Dokument wird das HTTPs WEBSERVICE Interface der WEBWARE (WW SVC) definiert und beschrieben.

Begriffe

WW-SVC	WEBWARE Services (WEB Services der WEBWARE)
WW-KIS	WEBWARE Kunden Installation
WW-SEAS	WEBWARE SoftENGINE AppStore
HTTP	Protokoll für den Zugriff und Datenaustausch zwischen Client und WW-System.
ServicePunkt	Der Zugangspunkt WWSVC erweitert die Domain https://myDomain.de/WWSVC
Hersteller	Erstellt Anwendungen, die über WW-SVC Schnittstelle ausgeführt werden. Der Hersteller wird mit einer eindeutigen 32-Byte-Hash-ID sowie Name definiert.
Anwendung	Eine Anwendung besteht aus einer definierten Funktionsgruppe und Rahmenparametern. Eine Anwendung wird mit einer eindeutigen 32-Byte-Hash-ID definiert.
SecuredApp	Eine Secured App wird durch Freigabe/Aktivierung einer Hersteller/Anwendung auf einem WW-System erstellt. Diese enthält Rahmenparameter/Zugriffsbeschränkungen und wird mit einer Secure-APP-ID (Zahl) zusätzlich zu Hersteller/Anwendung beschrieben.

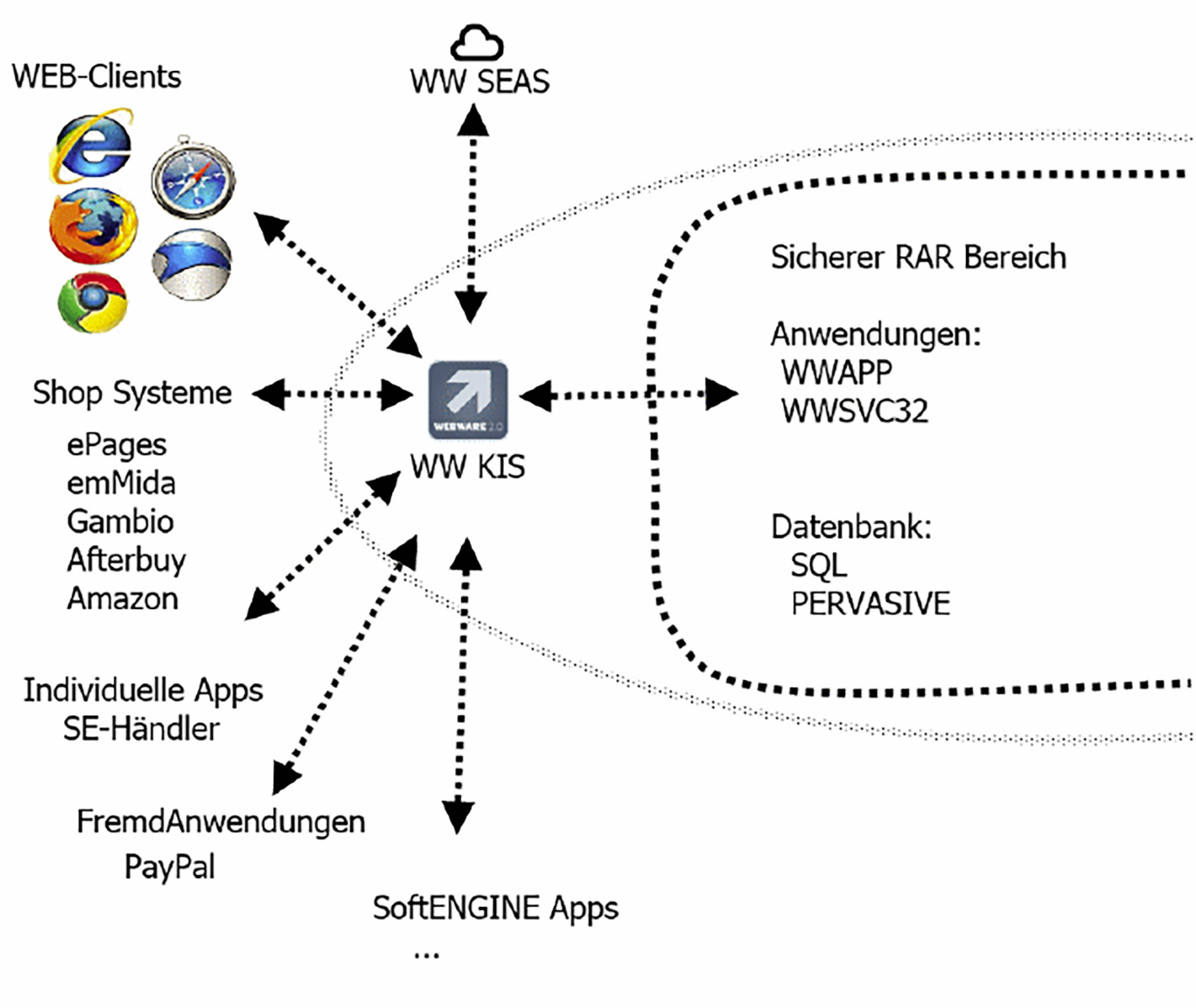
ServicePass	Möchte ein Client auf eine SecuredApp zugreifen, so muss er zuerst einen ServicePass beantragen. Der ServicePass besteht aus einem öffentlichen (PASS-ID) und geheimen (APP-SECRET) Hash-Wert.
PASS-ID	Dies ist ein öffentlicher HASH-Wert, der den Zugriff auf eine SecuredApp ermöglicht.
APP-SECRET	Dies ist ein geheimer HASH-Wert, der bei Registrierung an den Client übergeben wird. Dieser HASH-Wert wird danach nicht mehr innerhalb der Kommunikation verwendet.
COOKIES	Ist eine Erweiterung des HTTP-Headers der notwendige / optionale Informationen enthält.
Funktionsgruppe	Ist die Zusammenfassung von Einzelfunktionen und definiert auf welche Funktionen und mit welcher Berechtigung zugegriffen werden darf.
Funktion	Eine Funktion ist mit einem Namen beschrieben und kann über ein Aufruf über den ServicePunkt ausgeführt werden ../WWSVC/[PASS-ID]/Funktions-Name..
Parameter	Parameter können direkt beim Aufruf in der URL (URL-Encoded) übergeben werden. ../WWSVC/[PASS-ID]/MeineFunktion/10001/SORT=KNDNR Parameter können auch direkt im Body der WWSVC-Anfrage übergeben werden.

JSON	JavaScript Object Notation: Datenformat im Browser-Bereich.
XML	Extensible Markup Language: Datenformat im Browser-Bereich.
BenutzerListe	Eine Liste von Benutzer-Namen und Passwörtern mit der der Zugriff bei Registrierung und oder Funktionszugriff geschützt werden kann.
iWWSVC.js	Entwickler-Interface Test-Implementierung für WWSVC mit Javscript . Test: https://meine-webware.de/svc-js/svcjs.html

Zugriffsarten für WEBSERVICES:

SYNCHRON	Ergebnis der Service-Funktion wird mit der gleichen Verbindung zurückgegeben
ASYNCHRON	Service-Funktion wird im System eingelastet. Das Ergebnis kann zu einem späteren Zeitpunkt mit Hilfe des erhaltenen Asychron-Result-Handle abgerufen werden.
ASYNCHRON_NO_RESULT	Service-Funktion wird im System eingelastet. Das Ergebnis wird nicht benötigt und daher nach Abschluss der Service-Funktion nicht vorgehalten.

WWSVC ein Überblick



Die WEBWARE WEB Services WWSVC erlauben den Zugriff auf Daten, Funktionen und Prozesse Ihrer WEBWARE Installation aus Anwendungen (WWSVC-Apps) von Dritt-Herstellern heraus.
Da die Unternehmensdaten / Funktionen / Prozesse sicherheitskritisch sind, wurde hier auf die Art und Form des Zugriffs unter Sicherheitsaspekten höchsten Wert gelegt.
Die Verwaltung von Hersteller/Anwendungen (WWSVC-Apps) wird im SoftENGINE AppStore SEAS vom SoftENGINE Partner durchgeführt und konfiguriert.
WEBWARE Kunden-Installationen / WWKIS, welche WEB-Services nutzen wollen, registrieren sich direkt im SEAS und laden von dort ihre Hersteller/Anwendungen in ihre WWKIS herunter.
Der Zugriff auf WWSVC-Funktionen ist für Fremdanwendungen (Clients) nur über WWSVC-Apps möglich, die zuvor vom WEBWARE System Administrator der WWKIS als "Meine Service Anwendungen" freigegeben wurden.
Jeder Client der auf eine freigegebene "Service Anwendung" zugreifen will, muss sich hierfür einmalig registrieren und erhält dadurch einen Service-Pass mit dem er dann auf die bereit gestellten Funktionen/Prozesse/Daten der "Service Anwendung" zugreifen kann.
Die Abarbeitung der Service-Funktionen wird dabei skaliert im RAR-Bereich über die WWSVC32-Anwendungen durchgeführt.

WEBWARE iWWSVC.js

Beschreibung des Interfaces für WWSVC Javascript in der Version 1.03

Auslieferung

Sie finden die aktuelle Version des iWWSVC.js jeweils im aktuellen Setup/Update der WEBWARE 2.01 und 2.02 im folgenden Verzeichnis:

BIN\HOME\SVC-JS\

Zielbeschreibung

Das hier vorliegende Javascript Interface soll Ihnen helfen die WWSVC Schnittstelle der WEBWARE WEB Services (WWSVC) zu verstehen und im Browser anzuwenden.

Das iWWSVC.js speichert dabei notwendige Daten im Local-Storage des Browser um diese auch nach einem Browserneustart für Sie zur Verfügung zu haben.

Neben Hersteller/Anwendungen, Service-Pässen, Session-Token, Asynchron-Abfragen haben Sie auch die Möglichkeit, einfache Funktionen abzubilden und mit unterschiedlichen Aktionen auszuführen.

Aufbau der iWWSVC.js Schnittstelle

Das Interface besteht hierbei aus 3 Schichten:

Oberfläche svcjs-html/svcjs.css/testapp.js

Neben der HTML/CSS Datei für die Oberflächen Gestaltung finden Sie in der Datei „TESTAPP.JS“ die Anwendung der jWWSVC.js Schnittstelle als Beispiel.

iWWSVC.js WW-SVC-BASE.js

In dieser Datei sind die aktuellen Funktionen der iWWSVC.js dokumentiert sowie implementiert.

Interne WEBWARE Funktionen WW-SVC-MIN.JS bzw. WW-SVC-INT.JS

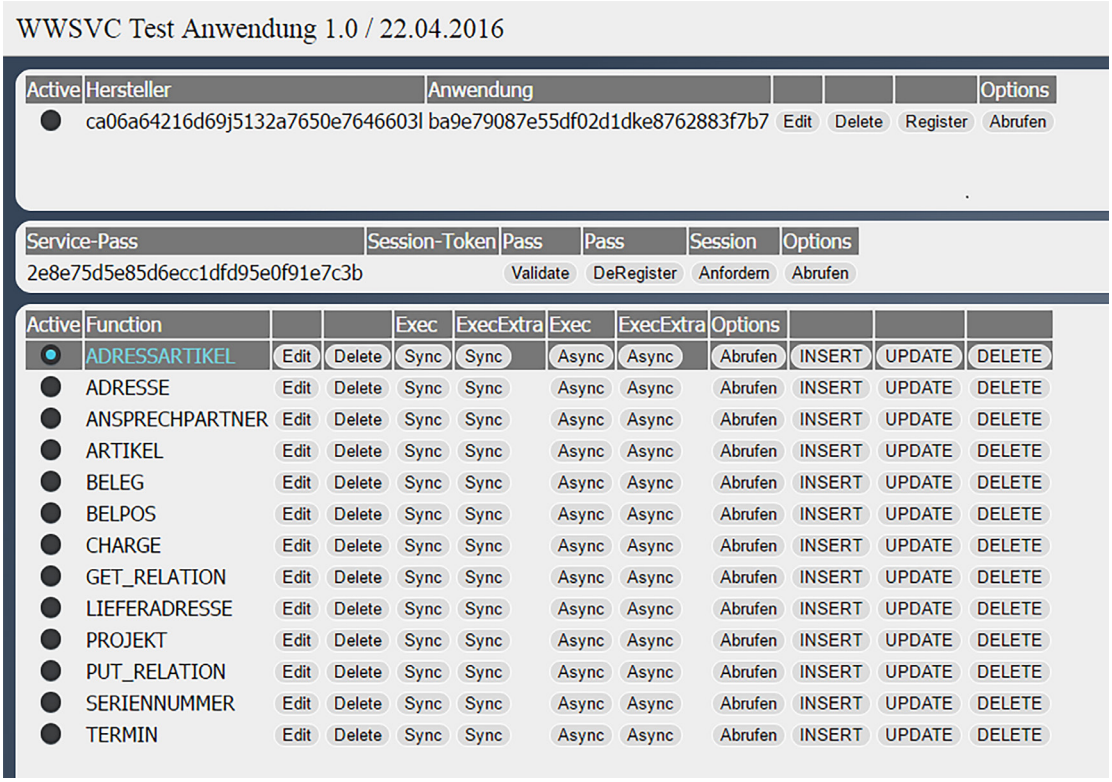
In der Datei WW-SVC-BASE.js sind die Schnittstellenfunktionen enthalten, die von Ihnen verwendet werden sollen. Dabei werden von der Datei WW-SVC-BASE.js die internen Funktionen aus der Datei WW-SVC-MIN.js gekapselt und mit einem einfachen Interface versehen. Hier sind also in einer Datei alle Tool und Grundfunktionen zusammengefasst, welche Low-Level für die Verwaltung von Service-Pässen, Hash-Wert Berechnung, Kommunikationsparameter Erstellung usw. notwendig sind.

iWWSVC.js Test-Anwendung / Oberfläche

Verwaltungs-Funktionen der Test-Anwendung

- mehrere Anwendungen
- je ein Service-Pass je Anwendung
- je ein aktueller Session-Token je Anwendung
- Eingabe von mehreren Funktionen für die Ausführung

Die Oberfläche wird durch drei Rahmen dargestellt, Anwendungen, Service-Pass und Funktionen. Durch Klick auf das PLUS-Symbol können Sie jeweils neue Elemente entsprechend des Rahmens anlegen.



Aktionen für bestehende Objekte können Sie innerhalb der Tabellendarstellung entsprechend auslösen. Beispiel: Validieren eines Service-Passes durch Klick auf die Schaltfläche "Validate" in der Service-Pass Zeile oder durch Session Anfordern ein Session-Token für einen Service-Pass anfordern.

In der Datei TESTAPP.JS finden Sie entsprechende Call-Back-Funktionen, welche durch Klick auf die Oberflächen-Controls ausgelöst werden.

Ein großer Teil der Test-Anwendung kümmert sich um die Speicherung und Ausführung von Funktionen, welche innerhalb der iWWSVC.js nicht benötigt werden.

iWWSVC.js Basis-Schnittstelle

Die Basis-Schnittstelle deckt mit 14 Grundfunktionen alle Bereiche des WEBWARE WEB Service (WWSVC) Systems ab.
Die Funktionen sowie die Ein- und Ausgangsparameter werden in der Datei WW-SVC-BASE.js genauer beschrieben. Die Beschreibung hier soll Ihnen einen kurzen Überblick geben.

Grundobjekte / Begriffe

AppHndl	Zugriffschlüssel auf eine interne Struktur mit Informationen zu einer Anwendung
ServicePass	Verwaltungsfunktion für einen Zugriffschlüssel einer Anwendung
SessionToken	Zugriffserlaubnis, die zeitlich begrenzt und per Benutzername / Passwort gesichert ist

Einleitung Beschreibung der Basis-Schnittstelle

Das hier verfügbare Javascript Interface für die WWSVC (iWWSVC.js) speichert für Sie die notwendigen Zustände und Laufzeitdaten.
Der Ablauf bzw. die Anwendung wird im Folgenden beschrieben:

A) Anwendungs-Verwaltung

Verfügbare Funktionen:

WWSVC_CONFIGURE_WW_SERVICE_APP (...)

WWSVC_GET_WW_SERVICE_APP_HANDLE (...)

WWSVC_GET_APPINFO (...)

WWSVC_DELETE_AI (...)

WWSVC_GET_APPLICATION_OPTIONS (...)

Anlagen bzw. Ändern einer Anwendungsbeschreibung

Rufen Sie für die Anlage bzw. zum Ändern von Anwendungsparametern zuerst diese Funktion:
WWSVC_CONFIGURE_WW_SERVICE_APP (WWSVC_SERVER_URL, WWSVC_SERVER_PORT, HERSTELLERID, ANWENDUNGSID, APP_SECURE_ID, APP_REVISION)

auf und übergeben Sie hier die notwendigen Informationen. Die iWWSVC.js speichert diese und gibt einen Anwendungs-Handle AppHndl zurück.
Wird die Funktion mit den Daten einer bestehenden Anwendung aufgerufen werden die Parameter aktualisiert. Es können mehrere Anwendungen in der iWWSVC.JS gleichzeitig gespeichert werden.
Falls Sie für eine bestehende Anwendung den Anwendungs-Handle, Informationen oder die Anwendung löschen wollen, stehen hierzu die drei übrigen Funktionen zur Verfügung:

Hole einen Anwendungs-Handle (AppHndl)

Hiermit können Sie einen AppHndl für eine bestehende Anwendung abfragen:

WWSVC_GET_WW_SERVICE_APP_HANDLE (HERSTELLERID, ANWENDUNGSID, APP_SECURE_ID, APP_REVISION)

Gebe die Beschreibung für einen AppHndl

Übergabe eines AppHndl and die Funktion, und man erhält eine Struktur mit den internen Beschreibungen

WWSVC_GET_APPINFO (AppHndl)

Lösche die Anwendungs-Informationen für einen AppHndl

Mit der Übergabe eines AppHndl an diese Funktion wird die interne Struktur für eine Anwendung gelöscht

WWSVC_DELETE_AI (AppHndl)

Abrufen der aktuellen WWSVC-Schnittstelle

Mit der Übergabe eines AppHndl an diese Funktion wird bei dem zugehörigen WWSVC-Server die JSON-API-Beschreibung für die ausführbaren Funktionen abgerufen. Sie erhalten dabei ein JSON-Objekt mit Funktionen und deren Eingangs und Ausgangsparametern.

WWSVC_GET_APPLICATION_OPTIONS (AppHndl, αCB)

B) Verwaltung von Service-Pässen

Für den Zugriff auf die WWSVC benötigen Sie neben der Anwendungs-Information einen ServicePass, der den Zugriff auf die WEBWARE SVC absichert. Hierzu stehen Ihnen drei Funktionen zur Verfügung, die es erlauben, für eine Anwendung einen ServicePass zu

Registrieren

bei Vorgabe von WWSVC-Server auch mit Angabe von Benutzer/Passwort
WWSVC_SERVICEPASS_REGISTER (AppHndl, aCB, opt_User, opt_PassWord)

Prüfen

ob der Service-Pass erlaubt ist, bzw. mittlerweile freigegeben wurde
WWSVC_SERVICEPASS_VALIDATE (AppHndl, aCB)

Löschen / Entfernen

also den Service-Pass beim WW-Server zu entfernen und dann im Browser die interne Service-Pass Struktur zu löschen
WWSVC_SERVICEPASS_REMOVE (AppHndl, aCB)

C) Anforderung eines Session-Token

Je nach Vorgabe im WWSVC-Servers kann die zusätzliche Authentifizierung mit Hilfe eines Session-Token erforderlich sein. Hier stehen Ihnen zwei Funktionen zur Verfügung

Neuer Session-Token anfordern

WWSVC_CONNECT_SESSION (AppHndl, UsrNam, UsrPW, aCB)

Vorhandener Session-Token löschen

WWSVC_DISCONNECT_SESSION (AppHndl, aCB)

D) Ausführen von WWSVC Funktionen

Für die Ausführung einer Funktion ist die Vorrausetzung, dass Sie einen gültigen ServicePass für den Zugriff auf den Ziel-WWSVC Server haben. Falls als erweiterte Anforderung die Benutzung von Session-Token erforderlich ist, so muss dieser ebenfalls vorhanden sein.
Es gibt 2 Funktionen, die Sie für die Ausführung der WWSVC-Funktionen mit iWWSVC.js für den direkten Aufruf verwenden können:

1) Aufruf mit Einzel-Angabe von Parametern

Hierbei werden die Funktionsparameter einzeln übergeben.

WWSVC_CALL_FUNCTION (AppHndl,FuncName, ParamValues, aCB,
opt_FuncVer, opt_Async, opt_VariableNameArray,
opt_VariableTypeArray, opt_AsyncPollInSecond,
opt_AsynchMaxTimeToPoll)

2) Aufruf mit Hilfs-Funktionen

Um die Funktionsparameter sowie Synchron/Asynchron Parameter korrekt übergeben zu können, gibt es zwei Hilfs-Funktionen, mit denen diese entsprechend formatiert werden:

Erstelle Funktionsparameter

WWSVC_HELPER_BUILD_FUNC_PARAM (FuncName, ParamValueArray, opt_FuncVer, opt_ParamNameArray, opt_ParamTypeArray)

Erstelle Asynchron-Parameter

WWSVC_HELPER_BUILD_ASYNC_PARAM (ExecAsync, AsyncPollInSecond, AsynchMaxTimeToPoll)

Die Ausführung erfolgt dann mit der Funktion

WWSVC_CALL_SVC_FUNCTION (AppHndl, aCB, FuncParam, opt_AsyncParam, opt_HTTPMode)

in dem Sie an diese die Ergebnisse der Hilfs-Funktionen sowie AppHndl und aCB übergeben.

.

3) Abruf von Optionen für Datenressourcen/Funktionen

Wenn Sie an diese Funktion einen AppHndl sowie einen Funktionsnamen übergeben, so erhalten Sie für das Objekt eine JSON-API Struktur mit der Beschreibung der Funktion und deren Parameter.

WWSVC_GET_FUNCTION_OPTIONS (AppHndl, FuncName, aCB, opt_FuncVer)

Wenn Sie an diese Funktion einen AppHndl sowie einen leeren Funktionsnamen übergeben, so erhalten Sie für die Liste der Ressourcen für die Anwendung übergeben.

WWSVC_GET_FUNCTION_OPTIONS (AppHndl, null, aCB, opt_FuncVer)

Wenn Sie an diese Funktion anstatt des Funktionsnamen den Namen einer Ressource übergeben, erhalten Sie hier die Liste der Funktionen für die Resource.

WWSVC_GET_FUNCTION_OPTIONS (AppHndl, „BELEGE“, aCB, opt_FuncVer)

E) Beschreibung der erwarteten CallBack-Funktion

Da die meisten Aktionen asynchron ausgeführt werden, d. h. das Programm wartet auf die Antwort des WWSVC-Servers, würde die Anwendung bis zum Eintreffen der Antwort "einfrieren". Daher werden bei einigen Funktionsaufrufen sogenannte Call-Back-Funktionen (Rück-Ruf-Funktionen) mit übergeben, welche nach Eintreffen des Ergebnisses ausgeführt werden.

Bis auf die Hilfsfunktionen zum Füllen von Funktionsparametern werden bei allen iWWSVC.js-Funktionen die Übergabe von Call-Back Funktionen erwartet. Wird diese nicht übergeben so werden die Aktionen trotzdem durchgeführt, jedoch erhält man dann keine Rückmeldung über den Erfolg bzw. die Daten von ausgelösten Funktionen

Hier die Beschreibung des Parameters aCB, der an diese iWWSVC-sj Funktionen übergeben wird. Es ist zu Beachten, dass je nach Funktionsaufruf nicht alle Felder gefüllt werden.

aCB : CallBack-Funktion mit Parametern CBP welcher aus folgenden Einträgen besteht

CBP.RESULTCODE	0 Fehler
	1 OK
CBP.ERRINFO	Beschreibung Rückgabe-Code/Zustand/Fehlerinfo
CBP.HTTPSTATUS	HTTP Status-Code
CBP.HEAD	COMESULT Objekt das zurückgegeben wurde
CBP.OBJ	Komplettes JSON Objekt aus der Rückgabe
CBP.BODY	Original HTTP-Body der empfangen wurde
CBP.MODE	Hier wird 'R' für Registrierung gesetzt

Um die Verwendung von Call-Back-Funktionen zu verstehen, sehen Sie diese angewendet in der TESTAPP.JS.

WEBWARE SVC WEBSERVICES

Wie sieht ein Zugriff auf WWSVC aus?

Folgende Aufgaben sind für einen Erstzugriff notwendig

- Administrator gibt eine Hersteller / App als SecuredApp für eine WW-Server-Instanz frei
- Client registriert sich über den WWSVC-ServicePunkt für die SecuredApp
- WW-Server akzeptiert die Registrierung und übergibt PASS-ID und APP-Secret
- Automatische Freigabe, oder WW-Server Administrator gibt gegeben falls den Zugriff manuell frei
- Client speichert lokal PASS-ID und APP-Secret
- Start Anwendungszugriff....

Notwendige Aufgaben für einen registrierten Client

- Client Sendet EXEC Anfrage an WWSVC-ServicePunkt mit PASS-ID und Funktionsanfrage
- Client Sendet bei Bedarf eine CONNECT-Anforderung falls Zugriff mit Benutzer/Passwort aktiviert ist.
- WEBWARE-Server bearbeitet die Anfrage und sendet Antwort

Zugriffsmodus Synchron/Asynchron

In der Basiskonfiguration der WEBWARE Services werden alle Anfragen SYNCHRON ausgeführt. Dabei wird die Verbindung beim Start einer Service-Funktion solange beim WEBWARE-Server gehalten, bis ein Ergebnis für die Service-Funktion vorliegt.

Um Ihnen und damit Ihren Anwendungen bei der Verwendung von Service-Funktionen mehr Flexibilität zu verschaffen, ist es auch möglich Service-Funktionen asynchron abzusetzen, wobei dann die Verbindung direkt nach der erfolgreichen Einlastung der Service-Funktion mit einem zugehörigen Asynchron-Handle zurückgegeben wird. Es ist dann möglich, zu einem späteren Zeitpunkt mit Hilfe des Asynchron-Handle das Ergebnis der Service-Funktion abzurufen. Desweiteren steht ein weiterer Modus zur Verfügung, mit dem kein Abrufen des Asynchronen Ergebnisses vorgesehen ist.

/WWSVC der ServicePunkt

Der Zugriff auf den Service-Zugang der WEBWARE erfolgt über einen definierten Service-Punkt mit der Kennung WWSVC. Alle Anfragen, die über diesen Service-Punkt erfolgen, werden in der Folge als WEBSERVICE-Anfrage interpretiert.

Der Service-Punkt bildet sich aus der WEBSERVER-Domain sowie /WWSVC. Da der WEBWARE-Server mehrere Internet-Zugangspunkte für mehrere WEBWARE-Instanzen anbietet, stehen die Service-Punkte damit für jede Instanz zur Verfügung.

Beispiel: https://MeineDomain.de/WWSVC
https://MeineDomain.de:4443/WWSVC

/WWSVC/WWSERVICE Funktionen Schnittstellenbeschreibung

Der /WWSVC-ServicePunkt bietet unterschiedliche Grundfunktionen an, mit denen ein Client mit dem WEBWARE-Server kommunizieren kann. Durch Angabe der WWSERVICE-Funktions-Kennung wird die Kommunikation direkt mit dem WW-Server aufgenommen.

Hier stehen Funktionen für die Registrierung, Prüfung, Löschung, Validierung eines Service-Passes und Verbindungs-Ende zur Verfügung. Dabei ist es teilweise notwendig weitere Funktionsparameter bzw. Cookies mit zu geben.

REGISTER

Ein Endgerät registriert sich als Client beim WEBWARE-Server für eine Secured-App. Hierbei wird Hersteller/Anwendung und Optional Benutzer/Passwort mit gegeben. Ist die Registrierung erfolgreich wird ein Service-Pass ausgestellt und zurückgegeben. Der Service-Pass ist nur dann sofort gültig wenn keine Freigabe durch den WEBWARE-Administrator notwendig ist.

VALIDATE

Prüfe ob ein Service-Pass für den Zugriff verwendet werden kann. Hierzu wird der Service-Pass übergeben und der Aufrufer erhält die Info ob der Service-Pass gültig ist.

DEREGISTER

Wird ein Service-Pass nicht mehr benötigt, so kann mit der DEREGISTER-Funktion ein Service-Pass entfernt werden.

CONNECT

Je nach Vorgabe des WEBWARE-Administrator für eine Secured-App kann es erforderlich sein, das bei einem Zugriff mit einem Service-Pass eine Authentifizierung, also die Anmeldung mit Benutzer und Passwort notwendig ist. Hierzu dient die Connect-Funktion. Bei dieser wird neben dem Service-Pass auch Benutzer und Passwort übergeben, welches vom WEBWARE-Server anhand von Benutzer-Listen für den Service-Pass ausgewertet werden.

Ist der Zugriff erlaubt, so wird ein Session-Token erstellt, welcher den Zugriff für eine bestimmte Zeit auf den Service-Pass erlaubt und der bei jeder Anfrage mit gegeben werden muss.

Erfolgt ein Zugriff auf einen Service-Pass ohne gültigen Session-Token, auch Erst-Zugriff, so wird die Fehlermeldung / Fehlercode 401 Authorization required (Anmeldung erforderlich)

CLOSE

Mit der CLOSE-Funktion kann ein Session-Token entwertet werden, also die Zugangserlaubnis für diesen Service-Pass aufgehoben werden. Ebenso wird damit der Zustand aktiv im System aufgehoben.

GETASYNCRESULT

Mit dem GetAsyncResult-Zugriff kann das Ergebnis für eine asynchron aufgegebene Service-Funktion abgeholt werden. Um eine Service-Funktion Asynchron abarbeiten zu lassen, muss diese eine Cookie mit der Kennung WWSVC-EXECUTE-MODE: ASYNCHRON, bzw. in der JSON-Struktur für den Service-pass die Variable "EXECUTE_MODE":"ASYNCHRON" setzen. Näheres hierzu weiter unten.

OPTIONS

Mit dem OPTIONS Befehl können Sie eine JSON-API-Beschreibung aller verfügbaren Schnittstellen Funktionen abrufen. Diese wird aus der Datei bin\wws\wwsvc\WWSVC-JSON.API gelesen.

Globale Rückgabe Information: COMRESULT

Wird ein Service-Pass bearbeitet und die Rückgabe ist keine binäre Datei, so erhält man ein JSON-Objekt mit Informationen über den Verarbeitungszustand und das Ergebnis zurück. Die COMRESULT Struktur hat folgende Felder:

- COMRESULT
  - STATUS Entspricht dem HTTP-Status-Code (200=OK)
  - CODE Entspricht dem HTTP-Status-Info („200 OK“)

Es folgen Optionale Felder, die teilweise über System-Wert (Detail-Level) aktiviert werden können.

- [INFO Text-Information über die Verarbeitung]
- [ERRORCODE Fehler/Hinweis-Code für das Ergebnis]
- [ERRORLINK WEB-Link auf die Dokumentation zu dem Error-Code]
- [ERRORINFO Genauere Fehler-Informationen, zur Fehleranalyse]
- [WWSVC_ASYNCHRON_HANDLE]
  Asynchroner Service-Handle bei Asynchron..

Werte in []-Klammern sind Optional.

Beispiel:

```
{        "COMRESULT":
        {
                "STATUS": 406,
                "CODE": "406 Not Acceptable",
                "INFO": "REGISTER is forbidden",
                "ERRORCODE": 10001,
                 "ERRORLINK":  "DOCWWSVC/ERR.HTML/#01001",
                "ERROINFO": "WWSVC IS NOT ACTIVATED, CALL ADMINISATRATOR"
        }
}
```

WWSVC/WWSERVICE - Funktionen im Detail

Registrierung eines Client /WWSVC/WWSERVICE/REGISTER

Ein Endgerät registriert sich als Client beim WEBWARE-Server für eine Secured-App. Hierbei wird Hersteller/Anwendung und Optional Benutzer/Passwort mit gegeben. Ist die Registrierung erfolgreich wird ein Service-Pass ausgestellt und zurückgegeben. Der Service-Pass ist nur dann sofort gültig, wenn keine Freigabe durch den WEBWARE-Administrator notwendig ist.

ServicePunkt: https://MeineDomain.de/WWSVC/WWSERVICE/REGISTER/

Möchte sich ein Client neu bei einem WW-Server registrieren, so muss er hier folgende Parameter mit übergeben.

- HERSTELLER Hersteller-ID 32 Byte HASH-Wert
- ANWENDUNG Anwendungs-ID 32 Byte HASH-Wert
- SecureAppID Angabe der internen Secure-AppID
- Revision Optional Revision der Anwendung
- USER Optional Benutzer Name
- PASSWORT Optional Benutzer Passwort
- CLIENTINFO Beschreibung des Client, Text für Administrator (wer ist es)
- CLIENTSECRET Optional, je nach Client Referenz-Hardware-Informationen

Beispiel eines Aufrufs

../WWSVC/WWSERVICE/REGISTER/HERSTELLER/APPID/SecureID//S.MUELLER//Client../

und nun eine komplette HTTP-Anfrage

GET /WWSVC/WWSERVICE/REGISTER/53f69160a5b0b89136ba1c6390c1e5d1/04abf1c38b8522869f857dcffa3c5500/1//Test-User// HTTP/1.1
Host:: myDomain.de
Connection: keep-alive

Es werden keine Cookies benötigt.
Der WEBWARE-Server prüft dann ob eine SecuredApp mit Hersteller-ID und Anwendungs-ID sowie Secure-App-ID für diesen ServicePunkt verfügbar ist und wie der Zugriff darauf erlaubt ist.
Wird vom Administrator eine Benutzer-Liste für die Registrierung vorgegeben, so wird der Benutzer-Name sowie das Passwort gegen diese Benutzer-Liste geprüft.
Der Rückgabe-Code sowie die Rückgabe-Daten hängen dabei von der Prüfung der Registrierung sowie des Freigabe-Modus (Automatisch bzw. manuelle Freigabe) für die Secured-App ab.

Mögliche Rückgabe-Codes

404 WWSCVC Subsystem ist auf dem WW-Server nicht verfügbar
406 Register für Anwendung nicht möglich, meist Anwendung nicht bekannt
202 Registrierung ok, jedoch Freigabe von Admin ausstehend
200 Registrierung ok, es kann sofort mit dem Service-Pass gearbeitet werden

Registrierung abgelehnt

Ist die SecuredApp (HerstellerID/AnwendungID) nicht vorhanden oder aktuell für Registrierungen gesperrt, bzw. Benutzer/Passwort nicht in einer vorhandenen Benutzer-Liste enthalten, so erhält der Client den HTTP-Statuscode 406 Not Acceptable als Antwort, sowie ein JSON-Objekt mit weiteren Informationen.

```
HTTP/1.1 406 Not Acceptable
Content-Length: 94
Content-Type: application/json

Date: Tue, 17 Mar 2015 23:37:33 GMT
Server: WebWare PoWeR Server

{ "COMRESULT":
{"STATUS": 406, "CODE": "406 Not Acceptable", "INFO": "REGISTER is forbidden",
"ERRORCODE": 10001, "ERRORINFO": "WWSVCDOKU/ERR.HTML/#01001"}}
{ "COMRESULT":
        {       "STATUS": 406,
                "CODE": "406 Not Acceptable",
                "INFO": "REGISTER is not possible",
                "ERRORCODE": 50100,
                "ERRORLINK": "DOCWWSVC/ERR.HTML/#50100",
                "ERRORINFO": "APPLICATION NOT KNOWN"
}       }
```

Registrierung OK, Freigabe von Administrator ausstehend

Ist die SecuredApp (HerstellerID/AnwendungID) vorhanden, aber der Administrator hat diese auf "Freigabe durch Administrator" konfiguriert, so erhält der Client bereits den Service-Pass mit dem Hinweis das der Zugriff noch nicht möglich ist. Erst durch Freigabe durch den Administrator ist der Zugang möglich. Wird in der Folge mit diesem Service-Pass

Hierbei wird der Return-Code HTTP 202 Accepted zurückgegeben, mit dem Service-Pass als JSON-Objekt im Datenbereich.


```
HTTP/1.1 202 Accepted
Content-Length: 246
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:41:05 GMT
Server: WebWare PoWeR Server

{ "COMRESULT":
      {  "STATUS": 202,
         "CODE": "202 Accepted",
         "INFO": "REGISTER OK, WAIT FOR ADMIN RELEASE",
         "ERRORCODE": 10000,
         "ERRORLINK": "DOCWWSVC/INFO.HTML/#10000",
         "ERRORINFO": "REGISTER OK WAIT FOR ADMIN RELEASE"
      },
"SERVICEPASS":
         {"PASSID": "9ff37b3b234b45d811acdc50fd0f80da",
          "APPID": "cb530381bbbb18cd33923c433ef7ee5c",
          "PDATE": 20150622,
          "PTIME": 8190176 }
}
```

Registrierung OK

Ist die SecuredApp (HerstellerID/AnwendungID) vorhanden und mit automatischer Freigabe konfiguriert, so ist nach der Registrierung der sofortige Zugriff mit dem Service-Pass möglich.
Hierbei wird der Return-Code HTTP 200 OK zurückgegeben, mit dem Service-Pass als JSON-Objekt im Datenbereich.


```
HTTP/1.1 200 OK
Content-Length: 216
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:39:13 GMT
Server: WebWare PoWeR Server

{ "COMRESULT": {
        "STATUS": 200,
        "CODE": "200 OK",
        "INFO": "REGISTER OK"
        "ERRORCODE":  0,
},
"SERVICEPASS":
{
        "PASSID": "44305f615eadfaca901b60692eeed7f4",
        "APPID": "38af61d2aff033ece56ba16ae0cbf472",
        "PDATE": 20150318,
        "PTIME": 391374
}}
```

Der Client muss dann die Daten des Service Passes lokal speichern, so dass er diese beim Zugriff auf den WW-SVC ServicePunkt übergeben kann.

PASSID: Service-Pass ID für Zugriff auf den WWSVC-Service-Punkt

APPID: APP-SECRET Application Secret, für Bildung von Dynamischen HASH-Werten

PDATE+PTIME sind die Erzeugungs-Datum/Uhrzeit auf Seiten des WW-Servers. Diese werden später benötigt um dynamische Hash-Werte zu Erzeugen.

Validierung ServicePass: /WWSVC/WWSERVICE/VALIDATE

ServicePunkt: https://MeineDomain.de/WWSVC/WWSERVICE/VALIDATE/ServicePassID

Hat ein Client einen ServicePass, und möchte prüfen ob dieser gültig ist, so kann er die VALIDATE-Funktion verwenden. Hierbei müssen neben dem Service-Pass auch die korrekten Validierungs-Cookie gesetzt werden.
Beispiel eines Aufrufs

../WWSVC/WWSERVICE/VALIDATE/123abcdServicePassIDrstruvxyz789

und nun eine komplette HTTP-Anfrage

GET /WWSVC/WWSERVICE/VALIDATE/0778e54d57f23b58bb0cdb2cb1e45691 HTTP/1.1
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
WWSVC-HASH: 88e49c05f9c49d86a2465117ec7e9de1
WWSVC-REQID: 14
WWSVC-TS: Tue, 17 Mar 2015 23:33:57 GMT

Mögliche Rückgabe-Codes

404 WWSCVC Subsystem ist auf dem WW-Server nicht verfügbar bzw.

404 Service-Pass nicht bekannt

202 Service-Pass ist OK, jedoch Freigabe von Admin ausstehend

200 Serivce-Pass ist OK, es kann sofort mit dem Service-Pass gearbeitet werden

Validierung schlägt fehl

Die Validierung des ServicePasses schlägt fehl wenn einer dieser Punkte zutrifft:

- ServicePass ist nicht bekannt
- ServicePass ist gesperrt
- WWSVC-APPID passt nicht zum ServicePass
- Zugriff ist wegen Zeitvorgaben aktuell nicht möglich
- Zugriff ist wegen Netzbeschränkungen nicht möglich
- ServicePass ist abgelaufen

Es wird dann mit dem HTTP-Code „404 Ressource not found „geantwortet. Im Body wird dazu ein JSON-Objekt mit COMRESULT-Objekt zurückgegeben welche genauere Informationen über den Grund der Ablehnung enthält.

HTTP/1.1 404 Resource not found
Content-Length: 104
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:53:46 GMT
Server: WebWare PoWeR Server

```
{ „COMRESULT“:
        {       „STATUS“: 404,
                „CODE“: „404 Resource not found“,
                „INFO“: „ERROR ServicePass not known“,
                  „ERRORCODE“:  1002,
                „ERRORLINK“: „WWSVCDOKU/ERR.HTML/#01002“
}       }
```

Validierung OK

Wird der ServicePass als gültig erkannt so wird der HTTP-Code 200 OK zurückgegeben. Im Body der HTTP-Antwort wird dann ein JSON COMRESULT-Objekt mit den Standardinformationen für Code 200 zurückgegeben.

Beispiel für HTTP Rückgabe bei Validierung (Service-Pass nicht freigegeben Code 202)

HTTP/1.1 202 Accepted
Content-Length: 98
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:50:16 GMT
Server: WebWare PoWeR Server

```
{ "COMRESULT":
        {       "STATUS": 202,
                "CODE": "202 Accepted",
                "INFO": "SERVICEPASS WAITING FOR RELEASE"
}       }
```

Beispiel für HTTP Rückgabe bei Validierung (Service-Pass ist freigegeben Code 200)

```
HTTP/1.1 200 OK
Content-Length: 75
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:52:03 GMT
Server: WebWare PoWeR Server

{ "COMRESULT":
        {       "STATUS": 200,
                "CODE": "200 OK",
                "INFO": "SERVICEPASS OK"
}       }
```

De-Registrierung eines Client /WWSVC/WWSERVICE/DEREGISTER

ServicePunkt: https://MeineDomain.de/WWSVC/WWSERVICE/DEREGISTER/ServicePass

Möchte sich ein Client seinen ServicePass löschen, so kann er mit dieser Funktion den ServicePass für den Zugriff sperren lassen.
Diese Funktion macht dann Sinn wenn ein Client die Zugriffssoftware löscht, bzw. er den ServicePass für die SecuredApp nicht mehr benötigt.

../WWSVC/WWSERVICE/DEREGISTER/123abcdServicePassIDrstruvxyz789

und nun eine komplette HTTP-Anfrage

GET /WWSVC/WWSERVICE/DEREGISTER/3e16a4e8061f8122bb782c2c1a340f2f HTTP/1.1
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
WWSVC-HASH: b866ec8edafbd50a4bd9c506afc99cfa
WWSVC-REQID: 38
WWSVC-TS: Tue, 17 Mar 2015 23:53:42 GMT

Mögliche Rückgabe-Codes

404 WWSCVC Subsystem ist auf dem WW-Server nicht verfügbar bzw.
404 Service-Pass nicht bekannt
200 Service-Pass wurde erfolgreich entfernt

DeRegistrierung schlägt fehl

Die DeRegistrierung des ServicePasses schlägt fehl, wenn einer dieser Punkte zutrifft:

- ServicePass ist nicht bekannt
- WWSVC-APPID passt nicht zum ServicePass

Es wird dann mit dem HTTP-Code 405 Not Allowed geantwortet. Im Body wird dazu ein JSON-Objekt mit COMRESULT-Objekt zurückgegeben welche genauere Informationen über den Grund der Ablehnung enthält.

HTTP/1.1 404 Resource not found
Content-Length: 104
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:57:29 GMT
Server: WebWare PoWeR Server

```
{ „COMRESULT“:
        {       „STATUS“: 404,
                „CODE“: „404 Resource not found“,
                „INFO“: „ERROR ServicePass not known“,
                „ERRORCODE“: 50200,
                „ERRORLINK“: „DOCWWSVC/ERR.HTML/#50200“,
                „ERRORINFO“: „“
}       }
```

DeRegistrierung OK

Wird der ServicePass als gültig erkannt so wird der HTTP-Code 200 OK zurückgegeben. Im Body wird dann ein JSON COMRESULT-Objekt mit den Standardinformationen für Code 200 zurückgegeben. Der WW SVC löscht dann den ServicePass aus der Zugriffsliste, so dass keine weitere Kommunikation mit diesem Pass möglich ist.

```
HTTP/1.1 200 OK
Content-Length: 85
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:53:42 GMT
Server: WebWare PoWeR Server

{ "COMRESULT":
        {       "STATUS": 200,
                "CODE": "200 OK",
                "INFO": "SERVICEPASS DEREGISTERED"
}       }
```

Neue Sitzung mit Benutzer-Liste eines Client

/WWSVC/WWSERVICE/CONNECT

Der Zugriff auf die WEBWARE Service-Schnittstelle mit einem gültigen Service-Pass kann weiter eingeschränkt werden. Dabei muss vor dem Zugriff auf die Service-Schnittstelle für einen Service-Pass eine Benutzer-Authentifizierung erfolgen.

Hierzu kann der Administrator in der SecuredApp eine gültige Benutzer/Passwort Liste hinterlegen und Vorgeben das bei einem Zugriff auf die SecuredApp der Client sich mit Benutzer und Passwort Anmelden muss.

Die Anmeldung erfolgt dabei über die CONNECT-Funktion bei der neben Service-Pass auch Benutzer und Passwort mit übergeben werden müssen.

Wurde die Anmeldung erfolgreich durchgeführt, wird ein Session-Token mit Hilfe eines Cookies: WWSVC-SESSION-TOKEN=xxx zurückgeben. Dieser WWSVC-SESSION-TOKEN Cookie muss dann bei allen weiteren Zugriffen mit den Service-Anfragen mit übergeben werden.

ServicePunkt: https://MeineDomain.de/WWSVC/WWSERVICE/CONNECT/ServicePass/Benutzer/Passwort

Beipiel:

/WWSVC/WWSERVICE/CONNECT/123abcdServicePassIDrstruvxyz789/Peter.Wolf/Entchen39

GET /WWSVC/WWSERVICE/CONNECT/...ServicePass.../Peter.Wolf/Entchen39

Host: myDomain.de

Mögliche Rückgabe-Codes

406 Fehler kein gültiger ServicePass wurde übergeben

404 WWSCVC Subsystem ist auf dem WW-Server nicht verfügbar bzw. ServicePass nicht bekannt

401 Fehler bei Benutzer/Passwort Prüfung bzw. Daten fehlen

200 Sitzung wurde erfolgreich gestartet

Erfolgreiche Anmeldung

```
HTTP/1.1 200 OK
Content-Length: 85
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:53:42 GMT
Server: WebWare PoWeR Server
Set-Cookie: WWSVC-SESSION-TOKEN ="04432570@1"; secure; httponly

{ "COMRESULT":
        {       "STATUS": 200,
                "CODE": "200 OK",
                "INFO": "AUTHENTIFICATION OK "
        },
        {"SESSIONTOKEN":
                {       "REQUIRED": 1,
                        "WWSVC_SESSION_TOKEN": "...."
}       }       }
```

Bei weiteren Zugriffen muss dann der Session-Token mit geliefert werden. Hierzu gibt es die Möglichkeit den Session-Token als HTTP-Header

WWSVC-SESSION-TOKEN: *"88e49c05f9c49d86a2465117ec7e9de1"*

oder in der JSON-Struktur für den Service-Pass mit zu übergeben:

```
"WWSVC_PASSINFO":
{
        "SERVICEPASS":          "0778e54d57f23b58bb0cdb2cb1e45691",
        "APPHASH ":    "88e49c05f9c49d86a2465117ec7e9de1",
        " TIMESTAMP":           "Tue, 17 Mar 2015 23:33:57 GMT",
        " REQUESTID":           "42",
        "SESSION_TOKEN": "88e49c05f9c49d86a2465117ec7e9de1"
},  ....
```

Ablehnung der Anmeldung:

ServicePass ungültig

Wird kein ServicePass übergeben der gültig ist so wird der Fehlercode 406 NOT ACCEPTABLE zurückgegeben.

HTTP/1.1 406 Not Acceptable
Content-Length: 85
Content-Type: application/json
Date: Tue, 17 Mar 2015 23:53:42 GMT
Server: WebWare PoWeR Server

{ "COMRESULT":
{ "STATUS": 406,
"CODE": "406 Not Acceptable ",
"INFO": " SESSIONTOKEN ERROR: NO VALID SERVICEPASS"
} }

ServicePass noch nicht freigegeben

Falls der Service-Pass noch nicht vom Administrator freigegeben ist, so wird dies mit der Fehlermeldung

{ „COMRESULT“:
{ „STATUS“: 406,
„CODE“: „406 Not Acceptable „,
„INFO“: „SESSIONTOKEN ERROR: SERVICEPASS WAIT FOR ADMIN RE-LEASE“
} }

Benutzer/Passwort Fehler

Wird kein passender Benutzer oder Passwort übergeben bzw. ist die Anmeldung nicht möglich so wird der Fehlercode 401 zurückgegeben.

HTTP/1.1 401 Authorization Required
Content-Length: 85
Content-Type: application/json
Date: Tue, 17 Mar 2015 23:53:42 GMT
Server: WebWare PoWeR Server

```
{ "COMRESULT":
        {       "STATUS": 401,
                "CODE": "401 Authorization Required ",
                "INFO": "USER OR PASSWORD NOT KNOWN"
}       }
```

Beende die Verbindung eines Client /WWSVC/WWSERVICE/CLOSE

ServicePunkt: https://MeineDomain.de/WWSVC/WWSERVICE/CLOSE/ServicePass

Möchte ein Client die aktuelle Verbindung für einen ServicePass Beenden, so kann er dies mit der CLOSE-Funktion durchführen. Dabei wird auch ein etwa-ig vorhandener Session-Token (CONNECT-Funktion) widerrufen und ist dann nicht mehr gültig.
Es ist auch möglich die Verbindung "hart bzw. kalt" zu unterbrechen, jedoch können mit der CLOSE-Funktion, Ressourcen die der WW SVC verwendet so-fort freigegeben werden.

../WWSVC/WWSERVICE/CLOSE/123abcdServicePassIDrstruvxyz789

und nun eine komplette HTTP-Anfrage

GET /WWSVC/WWSERVICE/CLOSE /123abcdServicePassIDrstruvxyz789
Host: myDomain.de
WWSVC-TS:
WWSVC-REQID:
WWSVC-APPID:

Mögliche Rückgabe-Codes

406 Fehler kein gültiger ServicePass wurde übergeben
404 WWSCVC Subsystem ist auf dem WW-Server nicht verfügbar bzw. ServicePass nicht bekannt
200 Sitzung wurde erfolgreich gestartet

CLOSE OK

Der WW SVC Beendet alle Verbindungen für diesen ServicePass und gibt HTTP-Code 200 OK zurückgegeben. Im Body wird dann ein JSON COMRESULT-Objekt mit den Standardinformationen für Code 200 zurückgegeben.

HTTP/1.1 200 OK
Content-Length: 85
Content-Type: text/html
Date: Tue, 17 Mar 2015 23:53:42 GMT
Server: WebWare PoWeR Server

```
{ "COMRESULT":
        {       "STATUS": 200,
                "CODE": "200 OK",
                „INFO“: „CONNECTION CLOSED“
}       }
```

Ergebnis asynchroner Service-Funktion abrufen /WWSVC/WWSERVICE/GETASYNCRESULT

../WWSVC/WWSERVICE/GETASYNCRESULT/123...ServicePassID...789/Async-Result-Handle Aufruf von GETASYNCRESULT mit einer gültigen Serivce-Pass-ID sowie gültigem Async-Result-Handle.

Wurde eine Service-Funktion zur Asynchronen Verarbeitung an den WEBWARE-Server (WWSVC) übergeben, so erhält man dabei einen Asynchronen-Result-Handle.

Der Asynchrone-Result-Handle wird mit Hilfe eines Header-Tags:

WWSVC-ASYNCHRON-HANDLE: WWSVC-ASYNC-06DA94C8

sowie in der Service-Pass Struktur wird dabei auch eine Variable

"WWSVC_ASYNCHRON_HANDLE": WWSVC-ASYNC-06DA94C8

Mit diesem Asynchron-Handle kann man dann beim WEBWARE-Server zeitlich versetzt das Ergebnis der Abfrage abholen.
Dabei kann es zu folgenden Zuständen kommen:

- Bearbeitung ist abgeschlossen, Ergebnis steht bereit (HTTP - Code der Verarbeitung)
- Noch in Bearbeitung (HTTP-Code 202)
- Async-Result-Handle ist ungültig (HTTP-Code 400)

Mögliche Rückgabe-Codes

400 Funktion nicht verarbeitbar/nicht bekannt, bzw. Async-Result-handle nicht gültig

404 WWSCVC Subsystem ist auf dem WW-Server nicht verfügbar

202 Asynchron Auftrag wurde eingelastet und wir erhalten einen Asyn chron-Handle

200 Ausführung erfolgreich, Ergebnis wird übergeben Asynchrone Operation abgeschlossen

Beispiel einer Anfrage die Asynchron bearbeitet werden soll:

```
PUT /WWSVC/EXECJSON HTTP/1.1
Content-Length: 456
WWSVC-EXECUTE-MODE: ASYNCHRON
WWSVC-REQID: 2
Content-Type: application/json
WWSVC-HASH: 0832f01df9b4e00d4820d1618017dac0
WWSVC-TS: Wed, 05 Aug 2015 08:26:42 GMT
WWSVC-SESSION-TOKEN: 0eaa1c78947cb822006bf0b9791eb028

{   "WWSVC_PASSINFO":
    {     "SERVICEPASS":"45c92b5959a3dd760ced41a7db5748b5",
          "APPHASH":"0832f01df9b4e00d4820d1618017dac0",
          "TIMESTAMP":"Wed, 05 Aug 2015 08:26:42 GMT",
          "REQUESTID":2,
          "EXECUTE_MODE":"ASYNCHRON"
    },
    "WWSVC_FUNCTION":
   {      "FUNCTIONNAME":"GET_ARTIKEL",
           "PARAMETER": ...
} }
```

Beispiel einer Antwort auf eine Asynchrone Anforderung:

Dabei wird der Asynchron-Result-Handle als Header-Feld sowie in der COM-RESULT-Struktur übergeben

```
HTTP/1.1 202 Accepted
Content-Length: 158
Content-Type: application/json
Date: Wed, 05 Aug 2015 10:27:54 GMT
WWSVC-ASYNCHRON-HANDLE: WWSVC-ASYNC-045A9220

{ "COMRESULT":
        {       "STATUS": 202,
                "CODE": "ASYNCHRON-FUNCTION-ACCEPTED",
                "INFO": "ASYNCHRON FUNCTION ACCEPTED",
                "WWSVC_ASYNCHRON_HANDLE": WWSVC-ASYNC-045A9220
}       }
```

Beispiel: Abrufen des Ergebnisses einer Asynchron in Auftrag gegebenen Service-Funktion

Hierzu wird die hier beschriebene WWSVC-Service-Funktion

./WWSVC/WWSERVICE/GETASYNCRESULT/Pass-ID/Asynch-Result-Handle verwendet. Bei dieser muss dabei ein gültiger Service-Pass sowie ein gültiger Asynchron-Result-Handle übergeben werden.

GET /WWSVC/WWSERVICE/GETASYNCRESULT/45c92b5959a3dd760ced41a7db5748b5/WWSVC-ASYNC-045C16D0 HTTP/1.1
WWSVC-SESSION-TOKEN: 0eaa1c78947cb822006bf0b9791eb028
WWSVC-EXECUTE-MODE: SYNCHRON
WWSVC-REQID: 5
WWSVC-HASH: a5e2d674244363925deec710ad43070b
WWSVC-TS: Wed, 05 Aug 2015 08:28:53 GMT

Je nach Bearbeitungszustand der asynchron in Auftrag gegebenen Service-Funktion kann es hier nun folgende Ergebnisse geben:

Bearbeitung ist abgeschlossen

Bei Zugriff auf eine fertig bearbeitete Service-Funktion erhalten Sie das Ergebnis wie gewohnt zurück. Durch den einmaligen erfolgreichen Zugriff auf das Ergebnis wird der Asynchron-Result-Handle ungültig und weitere Abrufe mit diesem sind nicht mehr möglich.

```
HTTP/1.1 200 OK
Content-Length: 120
Content-Type: application/json
Date: Wed, 05 Aug 2015 10:28:08 GMT

{"COMRESULT": {"STATUS": 200, "CODE": "200 OK", "INFO": "OK"},

   "ARTIKELLISTE":
   {
      "ANZAHL": "0"
   }
}
```

Der HTTP-Rückgabestatus der asynchronen Service-Funktion wird als Ergebnis gesetzt.

Bearbeitung ist noch nicht abgeschlossen (HTTP-Code 202)

Ist die Bearbeitung der Service-Funktion noch nicht abgeschlossen, so erhalten Sie den Status-Code 202 mit weiteren Informationen in der COMRESULT-Struktur

```
HTTP/1.1 202 Accepted
Content-Length: 207
Content-Type: application/json

{
        "COMRESULT":
        {
                "STATUS": 202,
                "CODE": "202 Accepted",
                "INFO": "ASYNCHRON-SVF-IN-PROGRESS",
                "ERRORCODE": 202,
                "ERRORLINK": "DOCWWSVC/INFO.HTML/#00202",
                "ERRORINFO": "ASYNCHRON Service Function in Progress"
}       }
```

Sie haben damit die Möglichkeit zu einem späteren Zeitpunkt erneut mit dem Asynchron-Result-Handle zuzugreifen um das Ergebnis abzurufen.

Zugriff fehlgeschlagen (HTTP-Code 400)

Ist der Asynchron-Result-Handle nicht gültig, so erhalten Sie einen HTTP-Fehlercode 400 mit weiteren Informationen in der COMRESULT-Struktur. Dabei wird der zusätzliche Fehlercode (ERRORCODE:) 50500 angezeigt.

```
HTTP/1.1 400 Bad Request
Content-Length: 204
Content-Type: application/json

{
        "COMRESULT":
        {
                "STATUS": 400,
                "CODE": "400 Bad Request",
                "INFO": "ASYNCHRON-HANDLE-NOT-KNOWN",
                "ERRORCODE": 50500,
                "ERRORLINK": "DOCWWSVC/ERR.HTML/#50500",
                "ERRORINFO": "ASYNCHRON HANDLE IST NOT KNOWN"
}       }
```

/WWSVC/EXEC Funktion

Mit der WWSVC EXEC Funktion kann nun für ein ServicePass eine zugehörige Funktion ausgeführt werden. Dabei können nur Funktionen ausgeführt werden die in der Funktionsgruppe für die SecuredApp definiert sind.
Es sind je nach Aufrufformat 2 Zugriffsfunktionen definiert. /EXECURL und /EXECJSON

/EXECURL Aufruf mit Funktionsbeschreibung und Parametern in der URL

/EXECJSON Aufruf mit Funktionsbeschreibung und Parametern im Body als JSON

HTTP / REST Funktionen
Um Funktionen auszuführen haben Sie die Möglichkeit diese per normalen PUT/POST auszulösen. Dabei ist zu Beachten das der WWSVC-Server eingehende Funktionsnamen bei Zugriff auf eine Datenobjekt wie zum Beispiel ARTIKEL, BELEGE, (...) oder WORKFLOW je nach HTTP-Befehl auf festgelegte Funktionen routet. Wird bei einem Funktionsaufruf direkt eine Funktion für ein Datenobjekt angegeben so entfällt das weitere Routen.
Beispiel (zur Vereinfachung im EXECURL Format):
Wenn Sie eine Funktion ausführen auf das Datenobjekt ARTIKEL und geben dabei keine weitere Ausführungs-Funktion an wie zum Beispiel (ARTIKEL.UPDATE, ARTIKEL.INSERT, ARTIKEL.DELETE ARTIKEL.GET), dann wird anhand des HTTP-Befehls die Ziel-Funktion automatisch angefügt.

PUT /WWSVC/SVP/ARTIKEL ARTIKEL.UPDATE
POST /WWSVC/SVP/ARTIKEL ARTIKEL.INSERT
GET /WWSVC/SVP/ARTIKEL ARTIKEL.GET
INSERT /WWSVC/SVP/ARTIKEL ARTIKEL.INSERT
DELETE /WWSVC/SVP/ARTIKEL ARTIKEL.DELETE

Beachten Sie das bei Ausführung mit dem GET-Befehl teilweise keine Übergabe von JSON-Parametern aus Browsern erfolgt.
Sie können also für den Zugriff auf die /WWSVC/EXEC... Funktionen folgende HTTP-Befehle verwenden:

- PUT entspricht Datenobjekt . UPDATE
- POST entspricht Datenobjekt . INSERT
- GET entspricht Datenobjekt . GET
- INSERT entspricht Datenobjekt . INSERT
- DELETE entspricht Datenobjekt . DELETE
- UPDATE entspricht Datenobjekt . UPDATE
- EXEC entspricht Datenobjekt . EXEC (Ausführen von Workflows usw.)
- OPTIONS Abfragen der API-JSON Struktur für das angegebene Objekt

Hinweis bei Verwendung von Benutzer-Listen

Ist der Zugriff für eine Service-Anwendung per Benutzer/Passwort Liste geschützt und ist die Anmeldung bei Zugriff aktiviert, so muss vor dem ersten Zugriff ein Connect mit Übergabe von Benutzer/Passwort erfolgen, damit die Verbindung über einen WWSVC-SESSION-TOKEN Cookie geschützt werden kann.
Hier ist ebenfalls zu Beachten das dieser WWSVC-SESSION-TOKEN nach Vorgabe ablaufen kann und bei Zugriff mit einem abgelaufenen WWSVC-SESSION-TOKEN eine „401-Authorization Required“ Fehlermeldung zurückgegeben wird.

Hinweis für die Bearbeitungsart (Synchron/Asynchron)

In der Basiskonfiguration der WEBWARE Services werden alle Anfragen SYNCHRON ausgeführt. Dabei wird die Verbindung beim Start einer Service-Funktion solange beim WEBWARE-Server gehalten bis ein Ergebnis für die Service-Funktion vorliegt.

Um Ihnen und damit Ihren Anwendungen bei der Verwendung von Service-Funktionen mehr Flexibilität zu verschaffen, ist es auch möglich Service-Funktionen Asynchron abzusetzen, wobei dann die Verbindung direkt nach der erfolgreichen Einlastung der Service-Funktion mit einem zugehörigen Asynchron-Handle zurückgegeben wird. Es ist dann möglich zu einem späteren Zeitpunkt mit Hilfe des Asynchron-Handle das Ergebnis der Service-Funktion abzurufen. Desweiteren steht ein weiterer Modus zur Verfügung mit dem kein Abrufen des Asynchronen Ergebnisses vorgesehen ist.

Der Ausführungs-Modus für eine Service-Funktion kann entweder über ein HTTP-Header Feld

WWSVC-EXECUTE-MODE: ASYNCHRON,

bzw. in der JSON-Struktur für den Service-Pass die Variable

"EXECUTE_MODE":"ASYNCHRON"

gesetzt werden.

Folgende Werte sind hier möglich:

- SYNCHRON (Standard-Wert, bzw. wenn Wert nicht angegeben, bzw. wenn nicht erlaubt
- ASYNCHRON — Asynchrone-Verarbeitung anfordern
- ASYNCHRON_NO_RESULT — Synchrone-Verarbeitung ohne Möglichkeit Ergebniszugriffs

/WWSVC/EXECURL Funktion

Mit diesem Funktionsaufruf können Sie direkt den Funktionsnamen sowie die Funktionsparameter in der HTTP-URL verwenden. Dieser Aufruf benötigt kein Body im HTTP-Call.

Beispiel der URL-Parameter für einen Funktionsaufruf:

Aufbau des ServicePunktes:

WWSVC	Service-Punkt
EXECURL	Ausführen von Funktion mit URL-String
PASS-ID	ServicePassID für die SecuredApp
Funktionsname	Funktion die ausgeführt werden soll, muss Bestandteil der APP-Funktiongruppe sein
Parameter-Liste	Je nach Funktion werden hier die Aufrufparameter an die Funktion angegeben

Hinweis zur Verwendung und Angabe von Parametern:

Position/Parameter-Name im Aufruf

Um eine Zuordnung des Parameters zu dem Funktionsparameter der aufgerufenen Funktion zu machen, haben Sie 2 Möglichkeiten.

- Angabe eines Parameter-Namens sowie ein Ist-Gleich-Zeichen und Parameter-Wert PARA=Wert

- Direkt den Parameter-Wert ../SET_ARTIKEL_LAGER/1000001/“Buchen“/

Im ersteren Fall, also Angabe von Name=Wert, wird direkt der Funktionsparameter unabhängig von seiner Position im URL-Aufruf gesetzt. Also ist zum Beispiel der Parameter "PARA4" der 4. Parameter so wird dieser bei Aufruf: ../STOREIMAGE/PARA4="TEST"/DATEINAME=bildklein.jpg/ korrekt als 4. Parameter für die Funktion zugeordnet.

Im zweiten Fall, also keine Angabe eines Parameter-Namens, wird der enthaltene Wert direkt anhand seiner Position als Funktionsparameter zugeordnet.
Angabe von Texten und dem URL-Trenner und Gleichheitszeichen
Der URL-Aufrufstring wird mit Hilfe des URL-Trenners / unterteilt. Daher ist es nicht möglich in einen normalen Parameter diesen Trenner als Wert aufzunehmen. Ebenso ist es nicht möglich das Gleichheitszeichen das zur Zuweisung eines Wertes zu einem Funktionsparameter Namens dient anzugeben.

Beispiel Pfad-Angabe für Datei /BILDNAME=bilder/meinbild.png/DELETE

Beispiel Druck-Text /ADDPRINTTEXT/100/100/Gesamtwert der Dienstleistung = 423,00 EUR/

In diesem Fall ist es notwendig den Parameter-Wert in Anführungszeichen zu setzen um damit Texte die die Sonderzeichen / und = enthalten korrekt verarbeiten zu können.

Korrekte Verwendung:
Beispiel Pfad-Angabe für Datei /BILDNAME="bilder/meinbild.png"/DELETE

Beispiel Druck-Text /ADDPRINTTEXT/100/100/"Gesamtwert der Dienstleistung = 423,00 EUR"/

Aufruf /WWSVC/EXECURL

Die Aufrufparameter werden hierbei wie oben beschrieben übergeben.

GET /WWSVC/EXECURL/63daf57370a3faf9e71f4260155d01a9/TESTFUNC/PAR1=Value/PAR2="Val/t-12" HTTP/1.1

Connection: keep-alive

Pragma: no-cache

Cache-Control: no-cache

WWSVC-HASH: 49b7b75dac82a56d95889a5e9b76eb78

WWSVC-REQID: 46

WWSVC-TS: Wed, 18 Mar 2015 00:00:11 GMT

Antwort auf /WWSVC/EXECURL

Der WW-Server antwortet mit HTTP Statuscode 200 wenn die Funktion erfolgreich ausgeführt werden konnte.

```
HTTP/1.1 200 OK
Content-Length: 350
Content-Type: text/html
Date: Wed, 18 Mar 2015 00:00:11 GMT
Server: WebWare PoWeR Server

{ "COMRESULT":
        {       "STATUS": 200,
                "CODE": "200 OK",
                "INFO": "OK TESTFUNCTION, get Color-Values"
        },
{ "COLORVALUES":
        { color: "red", value: "#f00"},
        {color: "green",        value: "#0f0"},
        {color: "blue",value: "#00f"},
        {color: "cyan",value: "#0ff"},
        {color: "magenta",value: "#f0f"},
        {color: "yellow",       value: "#ff0"},
        {color: "black",value: "#000"}}
}
```

Falls die Funktion nicht ausgeführt werden kann, so gibt es in der COMRESULT-Struktur der Antwort jeweils die Info warum die Funktion nicht ausgeführt werden konnte.

HTTP/1.1 404 Resource not found
Content-Length: 116
Content-Type: text/html
Date: Wed, 18 Mar 2015 00:07:20 GMT
Server: WebWare PoWeR Server

```
{ "COMRESULT":
        {       "STATUS": 404,
                "CODE": "404 Resource not found",
                "INFO": "ERROR SERVICEPASS IS NOT ALLOWED TO RUN"
}       }
```

/WWSVC/EXECJSON Funktion

Mit der WWSVC/EXECJSON können Sie ebenfalls einen Funktionsaufruf machen mit der Besonderheit das alle Cookies und Parameter als JSON-Objekt übergeben werden. Der Aufruf sollte in dem Fall als HTTP PUT Befehl ausgeführt werden. Der WW-Server erlaubt jedoch auch eine GET-Funktion mit Datenübergabe im Body des HTTP-Request.

WWSVCEXEC JSON-Objekt für Funktionsaufruf

Folgende Parameter werden im JSON-Objekt erwartet. Alle Parameter werden in Großschreibung angegeben.

WWSVC_PASSINFO

SERVICEPASS	ServicePass für diesen Aufruf
APPHASH	Berechneter Hash-Wert mit App-Secret und Zeitstempel
TIMESTAMP	Zeitstempel
REQUESTID	fortlaufende Zugriffsnummer der Verbindung

* Optionale Parameter weiter unten beschrieben, falls in JSON angegeben, hat dieses Vorrang

[GET_RESULT_MAX_LINES	Ergebnis einschränken / Bei 0 wird WW-Systemwert gesetzt]
[GET_WWSVC_CURSOR	Cursor in Ergebnis]
[GET_RESULT_TYPE	Form der Rückgabedaten]
[APPID	Name der Client-Anwendung zur Information/Debugging]
[SESSION_TOKEN	Session Token für Zugriffssteuerung]
[EXECUTE_MODE]	Wie soll die Ausführung erfolgen (DEFAULT: SYNCHRON)

[..] Optionale Parameter weiter unten beschrieben, falls in JSON angegeben, hat dieses Vorrang

WWSVC_FUNCTION

FUNCTIONNAME	Funktionsname der ausgeführt werden soll
REVISION	[Optional] Verwende Revision, 0=Default
PARAMETER	Array of Functionparameter
POSITION	[Optional] Nummer des Parameter für Übergabe
PNAME	[Optional] Name für Parameter
PTYPE	[Optional] Type des Datentypes (S=String,N=Number)
PCONTENT	Inhalt des Parameter

Beispiel einer JSON-Anfrage

```
PUT /WWSVC/EXECJSON HTTP/1.1
Connection: keep-alive
Content-Length: 454
Content-Type: application/json
{
        "WWSVC_PASSINFO":
        {
                "SERVICEPASS":      "0778e54d57f23b58bb0cdb2cb1e45691",
                "APPHASH ":  "88e49c05f9c49d86a2465117ec7e9de1",
                " TIMESTAMP":       "Tue, 17 Mar 2015 23:33:57 GMT",
                " REQUESTID":       "42",
                "SESSION_TOKEN": "88e49c05f9c49d86a2465117ec7e9de1"
        },
        "WWSVC_FUNCTION":
        {
                "FUNCTIONNAME":  "HELLO_WORLD",
                "PARAMETER":
                [
                        {"POSITION": "1",  "PNAME":  "USERNAME",  "PCONTENT":
"Aexel Bachli"},
                        {"POSITION": "2",  "PNAME":  "LANGAUGE",  "PCONTENT":
"DEUTSCH"}
                ]
        }
}
```

Mögliche Rückgabe-Codes

400 Funktion nicht verarbeitbar/nicht bekannt, Nähere Hinweise in der COMRESULT

406 Fehler kein gültiger ServicePass wurde übergeben

404 WWSCVC Subsystem ist auf dem WW-Server nicht verfügbar bzw. ServicePass nicht bekannt

200 Ausführung erfolgreich

Cookies und Transport Sicherheit

Zwischen Client und Server wird bei Registrierung ein öffentlicher Schlüssel (PASS-ID) sowie ein geheimer Schlüssel (APP-Secret) ausgetauscht.

Der PASS-ID Schlüssel dient zur öffentlich Übergabe des Service-Passes der verwendet werden soll. Dadurch ist es möglich in einer Client-Anwendung mehrere Service-Pässe zu verwenden. Der Service-Pass wird bei EXEC Funktion immer als Teil der Aufrufparameter übergeben.

Der APP-Secret Schlüssel bleibt auf Seiten des Client's und dient als Basis zu Erstellung eines MD5-Hash-wertes.

In der aktuellen Version wird die Validierung der Nachricht durch einen einfache HASH-Wert-Erstellung sichergestellt. Dabei wird jede neue Nachricht mit einem neu berechneten HASH-Wert übergeben. Dieser berechnet sich aktuell aus dem APP-Secret + aktueller Zeitstempel der als Cookie WWSVC-TS übergeben wird.

Beispiel für notwendige WWSVC-Cookies die bei jeder HTTP-Anfrage (bis auf die Registrierung) mit angegeben werden müssen

WWSVC-HASH: 49b7b75dac82a56d95889a5e9b76eb78

WWSVC-REQID: 46

WWSVC-TS: Wed, 18 Mar 2015 00:00:11 GMT

Optional

WWSVC-APPID: "mobile Auftragserfassung"

WWSVC-ACCEPT-RESULT-TYPE: "JSON"

WWSVC-ACCEPT-RESULT-MAX-LINES:100

WWSVC-REQID	laufende Request-ID

Hier wird eine fortlaufende Nummer der Anfragen übergeben. Diese kann später auch vom WW-Server für Validierung durch Sequenz Überprüfung verwendet werden.

WWSVC-TS Aktueller Zeitstempel

Hier wird der aktuelle Zeitstempel übergeben der zum Zeitpunkt des Senden beim Client erstellt wurde. Dieser Zeitstempel wird zur Zeit verwendet um mit dem APP-Secret einen MD5-Hash zu Berechnen.

WWSVC-HASH Meldungs Hash für Validierung

Mit diesem 32-Byte MD5-Hash-Wert wird auf Seiten des WW-Servers geprüft ob diese Meldung zu dem Service-Pass gehört und ob diese gültig ist.
Die Berechnung erfolgt mit MD5(App-Secret+WWSVC-TS)

WWSVC-APPID Name für Client-Anwendung

Hier kann der Client einen Namen für seine Anwendung vorgeben.
Beispiel: WWSVC-APPID: „SE:mobiles eBanking“

WWSVC-PASSID Service-Pass optional HTTP-Header

Hier kann der Client bei Aufruf von EXECJSON auch den Service-Pass als Header-Information setzen, um die Funktionsaufrufparameter nicht erneut erstellen zu müssen.
Beispiel: WWSVC-PASSID: „49b7b75dac82a56d95889a5e9b76eb78“

WWSVC-ACCEPT-RESULT-TYPE Form der Rückgabedaten

Hier kann man angeben, in welchem Format die Rückgabedaten gewünscht sind. Mögliche Formate sind:

- JSON JSON-Struktur ohne leere Zeilen/EndeLeerzeichen (DEFAULT)
- JSON-FILLED JSON-Struktur ohne Leerzeilen, Felder mit Leerzeichen aufgefüllt
- JSON-DETAIL JSON-Struktur mit leeren Zeilen und Felder aufgefüllt
- XML Es wird eine XML-Struktur zurückgegeben
- BIN * Rückgabe von Bildern/PDF usw. erwartet.
- DAT * noch nicht definiert

Beispiel: WWSVC-ACCEPT-RESULT-TYPE="JSON"

WWSVC-ACCEPT-RESULT-MAX-LINES Ergebnis einschränken

Wenn die Anzahl Ergebniszeilen nicht abzuschätzen sind, so kann man mit diesem Cookie die Anzahl des Ergebnisses beschränken. Standardmäßig geben alle Service-Funktionen alle gefundenen Sätze die zu der Anforderung passen zurück, [WWSVC-ACCEPT-RESULT-MAX-LINES=0 oder nicht vorhanden].
Da dies jedoch bei sehr großen Datenbeständen zu Laufzeitproblemen führen kann, (Bspl Abruf von allen Artikeln (500000..)) gibt es im System-Cockpit einen System-Wert mit dem bei keiner Vorgabe die maximale Anzahl der Sätze gesetzt werden.

Setze RESULT_MAX_LINES bei 0 auf	100	1	

Will man nur eine Bestimmte Anzahl von Ergebnis-Sätzen so kann man dies entsprechend mit WWSVC-ACCEPT-RESULT-MAX-LINES=123 (maximal 123 Stück, ähnlich wie bei SQL-Datenbank Top 123) eingrenzen.

Beispiel: Gebe mir alle Artikel aber, Maximal 123 Stück ..

"WWSVC-ACCEPT-RESULT-MAX-LINES": 123

WWSVC-CURSOR Cursorfunktionen

Mit dieser Vorgabe kann aus einem Ergebnis nur ein Teil abgerufen werden. Damit ist es zum Beispiel möglich ein sehr großes Result-Set in einer Tabelle darzustellen.
Sie können mit dem Befehl WWSVC-CURSOR solche großen Result-Set's verwalten. Wichtig ist hierbei die Angabe der Anzahl die maximal pro Zugriff zurückgegeben werden sollen (WWSVC-ACCEPT-RESULT-MAX-LINES=xx).
Wenn Sie einen Cursor erzeugen wollen, so wird dies mit folgender Angabe veranlasst:

WWSVC-CURSOR=CREATE
WWSVC-ACCEPT-RESULT-MAX-LINES=30

Werden wie hier im Beispiel mehr als 30 Ergebnis Sätze gefunden, so wird ein Cursor-Zugriff erzeugt und im Service-Punkt vorgehalten. Sie erhalten dann neben der ersten Ergebnissätze in der Rückgabe den Cursor-Zugriff in einem Cookie übergeben

WWSVC-CURSOR=[Kennung für Cursor, Installations-Abhängig..]

Wollen Sie die nächsten Sätze für einen offenen Cursor lesen, so erreichen Sie das mit der Angabe des Cursor-Zugriffs, sowie der gewünschten Ergebnis-Anzahl

WWSVC-CURSOR=[Kennung für Cursor, Installations-Abhängig..]
WWSVC-ACCEPT-RESULT-MAX-LINES=30

Werden hier weniger als die gewünschten Sätze gefunden, so wird der Cursor automatisch geschlossen, und Sie erhalten die Kennung WWSVC-CURSOR=CLOSED mit dem Ergebnis zurück.
Hinweis: Es ist zur Laufzeit immer nur ein Zugriff auf einen Cursor erlaubt, ist bereits für einen Cursor ein Service-Anfrage im System und noch nicht fertig bearbeitet, so wird ein erneuter Zugriff auf diesen Cursor mit einem Fehler quittiert.

WWSVC-SESSION-TOKEN

Wird für den Zugriff auf eine SecuredApp vom Administrator festgelegt, das der Zugriff per Benutzer-Liste und nur mit einem gültigen Session-Token erfolgen darf, so muss der Client diesen Session-Token mit dem Befehl CONNECT anfordern. (Siehe weiter oben)

/WWSVC/WWSERVICE/CONNECT

In der Folge muss dann bei jedem Zugriff dieser WWSVC-SESSION-TOKEN übergeben werden. Falls der WWSVC-SESSION-TOKEN ungültig wird (Zeitablauf, bzw. Zugriff des gleichen Service-Pass ohne WWSVC-SESSION-TOKEN) erhält der Client die Fehlermeldung das der WWSVC-SESSION-TOKEN neu angefordert werden muss, mit dem Status-Code 401 AUTHORIZATION REQUIRED.

WWSVC-EXECUTE-MODE

Mit dem WWSVC-EXECUTE-MODE kann angegeben werden ob eine Service-Funktion SYNCHRON (Standard) oder ASYNCHRON bearbeitet werden soll.
Bei Synchroner-Bearbeitung wird die verwendete Verbindung erst nach Abschluss der Bearbeitung der Service-Funktion mit dem entsprechenden Ergebnis an den Aufrufer zurückgegeben.
Bei langlaufenden Service-Funktionen kann es von Vorteil sein, diese Asynchron ausführen zu Lassen. Dabei wird direkt nach der Eingangsprüfung ein Asynchron-Handle zurückgegeben. Mit diesem Asynchron-Handle kann dann zu einem späteren Zeitpunkt das Ergebnis für die Service-Funktion abgerufen werden.
Ist das Ergebnis für die Service-Funktion nicht wichtig, bzw. soll zu einem späteren Zeitpunkt kein Ergebnis abgerufen werden, so kann mit der Vorgabe.

WWSVC-EXECUTE-MODE=ASYNCHRON_NO_RESULT

die Service-Funktion ohne Asynchron-Handle abgesetzt werden. Damit ist dann kein weiterer Zugriff für diese Service-Funktion mehr notwendig.
Mögliche Parameter für WWSVC-EXECUTE-MODE:

- SYNCHRON (*DEFAULT wird immer gesetzt wenn nicht angegeben/ unerlaubter Wert)
- ASYNCHRON
- ASYNCHRON_NO_RESULT

Es ist zu Beachten das der Asynchron-Modus nur erlaubt ist, wenn dies für die Service-Funktion bzw. das WWSVC-System entsprechend aktiviert wurde. Ist die Asynchrone Ausführung nicht erlaubt, so wird der Zugriff automatisch SYNCHRON ausgeführt.

Es ist möglich eine maximale Vorhaltezeit nach Abarbeitung der Service-Funktion für das Ergebnis vorzugeben. Dabei wird nach Erreichen der maximalen Vorhaltezeit der Asynchron-Handle ungültig und ein Abrufen des Ergebnisses ist dann für die Service-Funktion nicht mehr möglich.

Bei Asynchroner Bearbeitung wird ein Zugriffs-Schlüssel für den späteren Zugriff zurückgegeben. Dabei ist zu Beachten das hier jeweils ein Cookie WWSVC-ASYNCHRON-HANDLE, sowie ein Attribut in der COM-RESULT-JSON Struktur WWSVC_ASYNCHRON_HANDLE zurückgegeben wird.

Bei ASYNCHRON erhält man dabei den Zugriffs-Handle, bei ASYNCHRON_NO_RESULT wird der Standard-Wert ASYNCHRON-ACCEPTED zurückgegeben, da kein weiterer Zugriff mehr notwendig ist.

Übersicht der Fehler/Status-Codes

HTTP-Status-Codes

Folgende Status-Codes werden vom WWSVC im WW-Server verwendet. Dabei werden die vordefinierten Status-Codes von HTTP 1.1 verwendet.

Status - OK

Code	Info	Details
200	OK	Aktion wurde erfolgreich ausgeführt
202	ACCEPTED	Aktion wurde erfolgreich ausgeführt, es sind noch weitere Aktionen notwendig. Beispiel: Service-Pass wurde registriert, muss jedoch noch von Administrator freigegeben werden. Beispiel: Es wurde eine Asynchrone Anfrage abgesetzt (WWSVC_EXECUTE_MODE=ASYNCHRON), bzw. eine Asynchrone Anfrage ist noch nicht fertig bearbeitet.

Status-Fehler

Code	Info	Details
400	BAD REQUEST	Anfrage ist Fehlerhaft, wird abgelehnt
401	AUTHORIZATION REQUIRED	Es ist eine Anmeldung für den Zugriff erforderlich, der Client muss Benutzer und Passwort für die Secured-App angeben. In der SecuredAPP kann eine Benutzerliste festgelegt werden, welche bei Registrierung und/oder Zugriff geprüft wird. Ist eine Benutzerliste für den EXEC-Zugriff eingetragen, so wird bei jedem EXEC-Aufruf ohne gültigen SESSION-TOKEN der Code 401 zurückgegeben. Für einen gültigen SESSION-TOKEN ist ein CONNECT-Aufruf erforderlich.
403	FORBIDDEN	Zugriff auf diese Funktion ist nicht erlaubt
404	NOT FOUND	Die angeforderte Funktion ist nicht gültig. Beispiel: es wird mit einem Service-Pass auf die WEBWARE-Services zugegriffem, welcher nicht bekannt ist.
405		
406	NOT ACCEPTABLE	Die Anfrage kann nicht angenommen werden. Beispiel: Versuch der Registrierung eines Servicepasses, aber WEBWARE-Services sind deaktiviert
409	CONFLICT	Der Zugriff ist wegen eines Conflictes mit einem anderen Zugriff nicht möglich

Server-Status-Fehler

Code	Info	Details
500	INTERNAL SERVER ERROR	Bei der Bearbeitung der Service-Anfrage ist ein interner Fehler im WEBWARE-Server aufgetreten, die Anfrage konnte nicht bearbeitet werden.
501	NOT IMPLEMENTED	Die angeforderte Funktion ist vom WEBWARE-Service nicht implementiert

Liste der Rückgabe-Codes

Wird eine Service-Funktion ausgeführt, so kann im Fehlerfall eine Zusatzinformation ausgegeben werden, welche nähere Hinweise und Details zu dem Bearbeitungszustand enthält. Wenn Sie diese Codes erhalten wollen, so muss der Detailgrad für die COMRESULT-Struktur auf mindestens 1 gesetzt werden. Dieser kann im System-Cockpit unter aktiviert und gesetzt werden.

INFO 0 .. 9999

Code	Bereich	Details
0	ALL	Ausführung OK
200	GETASYNCRESULT	Ausführung OK, Ergebnis wird übergeben. Asynchron-Handle ist nun gültig.
201	VALIDATE	Servicepass wurde erfolgreich geprüft und Zugriff ist erlaubt
202	GETASYNCRESULT	Service-Funktion wird asynchron abgearbeitet. Ergebnis ist noch nicht verfügbar.

WARNUNGEN 10000 .. 49999

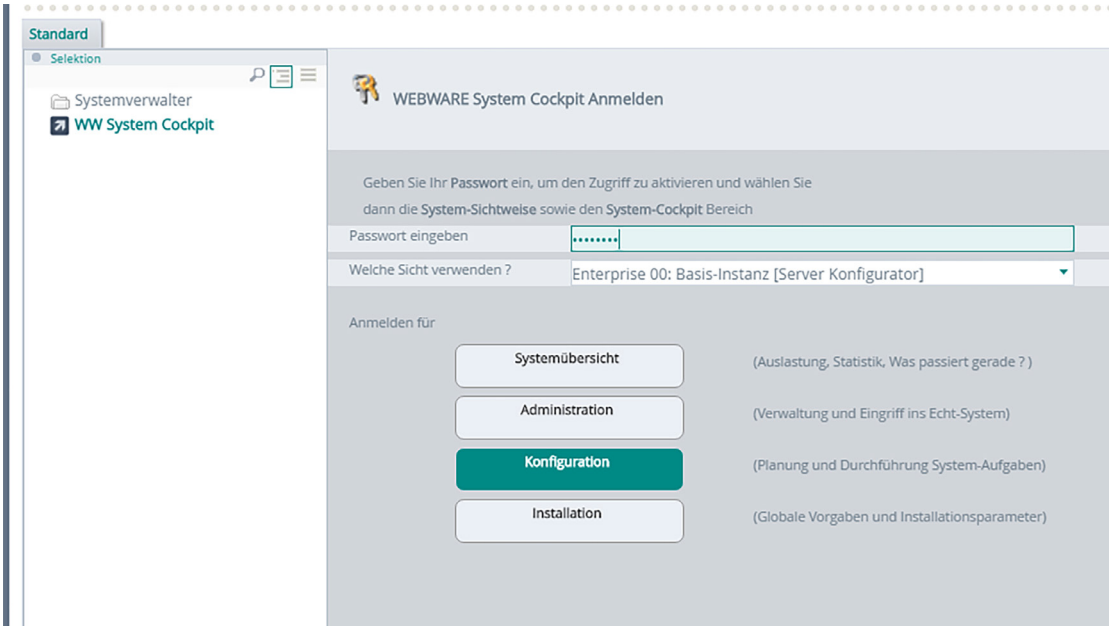
Code	Bereich	Details
10000	REGISTER	Service-Pass wurde registriert, wartet noch auf Freigabe von Administrator

FEHLER 50000 .. 99999

Code	Bereich	Details
50000	SERVER	Anfrage war nicht zu verarbeiten, Verarbeitung abgebrochen
50100	REGISTER	Servicepass nicht zu registrieren, da Anwendung nicht bekannt
50101	REGISTER	Servicepass nicht zu registrieren, da Anwendung für Registrierung gesperrt
50102	REGISTER	Servicepass nicht zu registrieren, da Zugriff aus diesem Netzsegment verboten ist
50103	REGISTER	Servicepass nicht zu registrieren, da für die SecuredApp eine Benutzerliste für die Registrierung aktiviert ist und kein Benutzer/Passwort, bzw. kein passendes übergeben wurde.
50200	VALIDATE	Servicepass nicht gefunden
50300	CONNECT	Fehler beim CONNECT mit einem Service-Pass, Benutzer/Passwort nicht in Benutzerliste vorhanden oder Service-Pass ist nicht lauffähig.
50301	CONNECT	Fehler Service-Pass ist nicht erlaubt
50302	CONNECT	Fehler Service-Pass ist nicht freigeschaltet.
50400	EXEC GETASYNCRESULT	Session-Token für Zugriff ist nicht gültig. Rückgabe Code 401 Authentifizierung ereforderlich (CONNECT..) um einen gültigen WWSVC-SESSION-TOKEN zu erstellen.
50500	GETASYNCRESULT	Zugriff ist nicht erlaubt. Asynchron-Handle sit nicht gültig

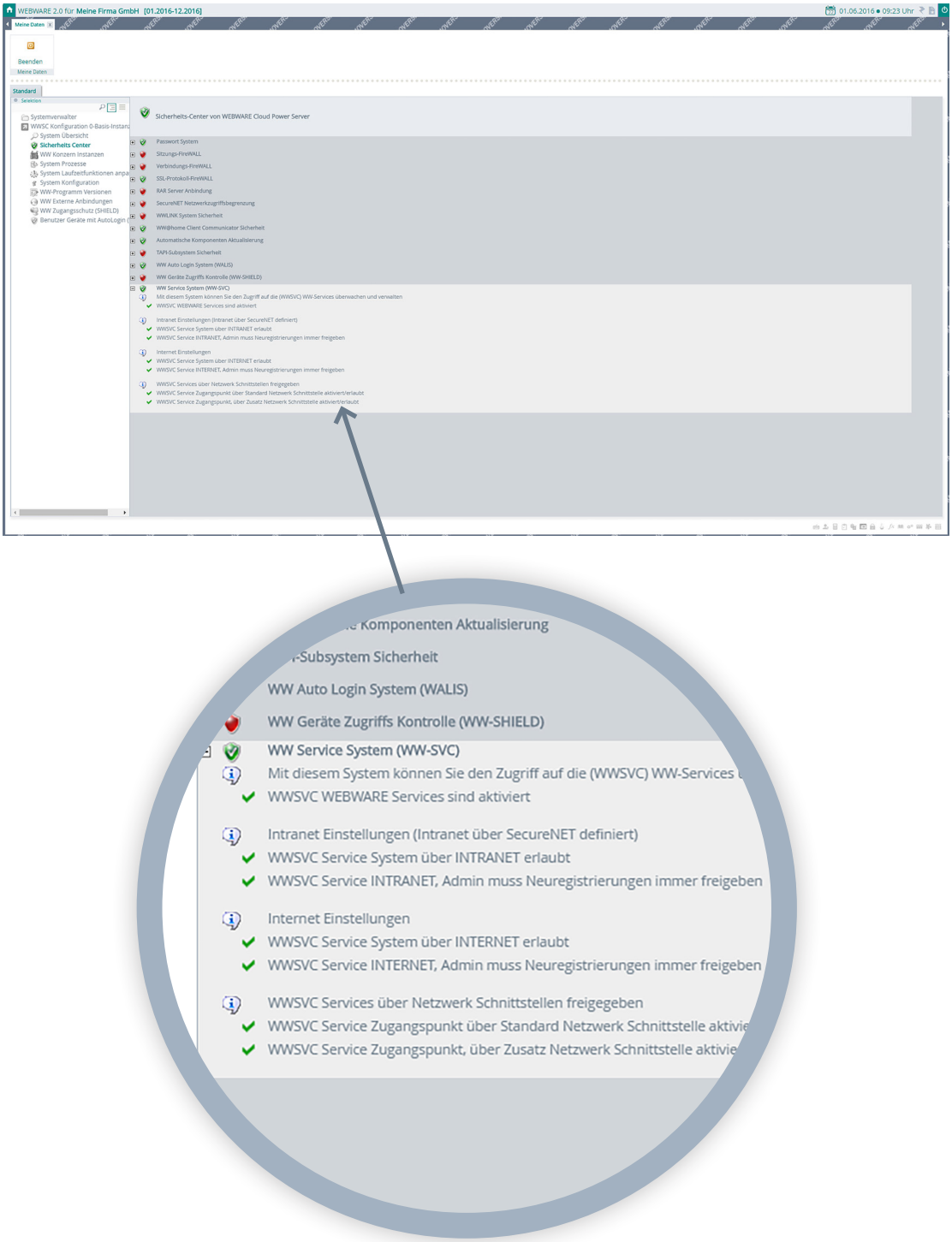
Aktivieren der WEBWARE Services im SystemCockpit

Um das WEBWARE Service System WWSVC für Ihr WEBWARE System zu aktivieren, melden Sie sich mit der Option Konfiguration an.

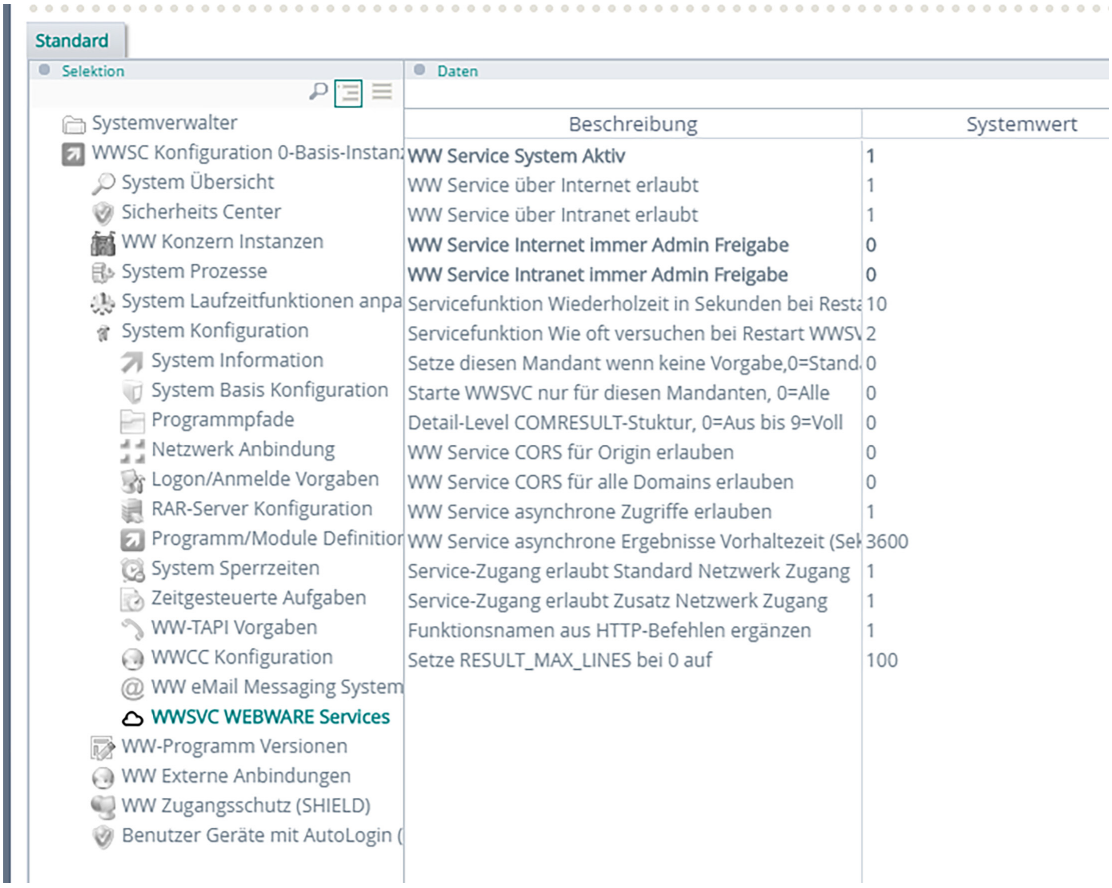


Aktivieren von WWSVC

Sie finden die wichtigsten Parameter zur Aktivierung der WEBWARE Services (WW-SVC) im Bereich Sicherheit-Center. Sie können hier mit dem ersten Parameter die WW-SVC aktivieren. Mit den 4 weiteren Parametern können Sie die Zugriffs- und Freigabevorgaben für Inter- und Intranet vorgeben.



Weitere Parameter finden Sie unter "System Konfiguration" in dem Eintrag WWSVC WEBWARE Services Einträge.



Die Bedeutung der Konfigurationseinträge

WW Service System Aktiv

Ist dieser aktiviert, so wird beim nächsten Start des WEBWARE-Servers das WW-SVC Subsystem mit hochgefahren, und ist verwendbar.

WW Service über Internet erlaubt

Ist dieser Schalter aktiviert, so ist der Zugriff mit Service-Pässen aus dem Internet heraus erlaubt. Vorgabe welcher Bereich Intranet und welcher Bereich Intranet ist wird in der Konfiguration im Bereich „Konfiguration“ > „System Konfiguration“ > „Login/Anmelde Vorgaben“ > „Inter-Net bzw. Intra-Net“ gemacht.

WW Service Internet immer Admin Freigabe

Meldet sich eine Service-Anwendung an Ihrem WEBWARE System an so haben Sie die Möglichkeit Zugriffe für die Service-Pässe auf Hersteller/Anwendungsebene ohne Freigabe durch Administrator freizugeben. Ist dieser Schalter hier gesetzt, so muss beim Erstzugriff von Service-Pässen aus dem Internet immer ein Administrator diesen nach der Registrierung freigeben.

WW Service über Intranet erlaubt

Mit diesem Schalter können sie den Zugriff auf WEBWARE Services aus dem Intranet heraus erlauben.

WW Service Intranet immer Admin Freigabe

Meldet sich eine Service-Anwendung an Ihrem WEBWARE System an so haben Sie die Möglichkeit Zugriffe für die Service-Pässe auf Hersteller/Anwendungsebene ohne Freigabe durch Administrator freizugeben. Ist dieser Schalter hier gesetzt, so muss beim Erstzugriff von Service-Pässen aus dem Intranet immer ein Administrator diesen nach der Registrierung freigeben.

Servicefunktion Wiederholzeit in Sekunden bei Restart

Wird eine Servicefunktion in einem WWSVCxx-Knoten für die Ausführung eingeplant, und kommt es zu einem Restart/Abbruch des WWSVCxx-Knoten, so kann hier die Zeit in Sekunden bis zum letzten Versuch die Servicefunktion erneut einzuplanen vergehen soll.

Setze diesen Mandant wenn keine Vorgabe, 0=Standard-Mandant

Falls keine Mandanten-Vorgabe beim Zugriff von Service-Pässen vorhanden ist, so kann hier ein Mandant angegeben werden, der für den Zugriff auf die WWSVCxx-Knoten verwendet wird.

Starte WWSVC nur für diesen Mandanten, 0=Alle

Bis die Konfiguration des WWSVC-Systems vollständig möglich ist, kann man mit diesem Schalter den Start der WWSVCxx-Knoten auf einen einzelnen Mandanten beschränken, bzw. mit 0 die WWSVCxx-Knoten für jeden Mandanten aktivieren. Diese Option ist ab Version 09.02.2016 ohne Funktion.

Detail-Level COMRESULT-Struktur 0=Aus bis 9=Voll

Hier kann angegeben werden wie viel Informationen / Hinweise in der COMRESULT-Rückgabe Struktur vom WEBWARE-Server WWSVC übergeben werden sollen.

- Level 0: (DEFAULT)

Rückgabe von STATUS, CODE, INFO

- Level 1:

Rückgabe von STATUS, CODE, INFO, ERRORCODE

- Level 2:

Rückgabe von STATUS, CODE, INFO, ERRORCODE, ERRORLINK

- Level 3:

Rückgabe von STATUS, CODE, INFO, ERRORCODE, ERRORLINK, ERRORINFO

weitere Level's werden später noch definiert.

Level 8: Zeit-Debug-Modus

Es wird eine Struktur mit Informationen über das Laufzeitverhalten mit zurück gegeben

Level 9: Voll-Debug-Modus

Es wird eine Struktur für die interne Funktionsabarbeitung im Prozessor-Knoten mitgegeben.

Was bedeuten die einzelnen COMRESULT-Werte

STATUS:	Status-Code (HTTP-Status-Code siehe Tabelle weiter oben)
CODE:	Standard-Beschreibung HTTP-Status-Code
INFO:	Zusätzlicher Beschreibender Text vom WEBWARE WWSVC System
ERRORCODE:	Interner Fehler/Hinweis/Warnungs-Code der auf die Dokumentation verweist siehe weiter oben in Liste der Rückgabe-Codes..
ERRORLINK:	Link auf die HTML-Dokumentation in Ihrem WW-Server zu dem ERRORCODE
ERRORINFO:	Zusätzlicher Beschreibender Text zu dem ERRORCODE

Kurze Einführung in CORS

Wird eine Verbindung von einem Browser zu Ihrem WEBWARE-Server System aufgebaut, und die Basisanwendung im Browser wurde nicht von Ihrem WEBWARE-System aufgerufen, so kommt es im Browser zu einem Fehler, da eine Verbindung mit Ajax und anderen Web-Technologien nur zu dem Ursprungs-Server von die Basisanwendung aufgerufen wurde erlaubt ist. Diese Browser-Interne Sicherheits-Restriktion kann mit Hilfe von CORS umgangen werden. CORS steht für Cross-Origin Resource Sharing, also frei übersetzt..“Zugriff auf Resourcen auch von anderen Ursprungs-Rechner erlauben“.

Bei CORS muss der aufgerufene WEB-Server (in diesem Falle Ihr WEBWARE-Server) die Erlaubnis und auch die Reichweite für diese Verbindung angeben. Dabei wird angegeben wie mit den Sitzungsparametern, Cookies (HTTP-Header) und den Daten umgegangen werden soll. Dürfen diese nur bei Verbindungen zu Ihrem WEBWARE-Server, oder auch bei anderen Verbindungen verwendet werden.

WW Service CORS für Origin erlauben

Ist dieser System-Wert aktiviert, so ist CORS nur für diese Verbindung zu Ihrem WEBWARE-Server gültig. Sitzungsparameter dürfen vom Browser nicht an andere Domains weitergegeben werden.

WW Service CORS für alle Domains erlauben

Ist dieser System-Wert aktiviert, so ist CORS für alle Verbindungen auch zu Ihrem WEBWARE Server gültig und die Parameter für die Verbindung zu Ihrem WEBWARE-Server dürfen auch mit anderen Systemen geteilt werden. (nicht so gut..)

WW Service ASYNCHRONE Zugriffe erlauben

Mit diesem System-Wert kann der Asynchrone Zugriff auf Ihre WEBWARE-Services aktiviert werden. Bei Asynchronen Zugriffen wird das Ergebnis des Zugriffs zwischengespeichert und kann später vom Initiatior der Service-Funktion abgerufen werden.

WW Service ASYNCHRONE Ergebnisse Vorhaltezeit (Sekunden)

Mit diesem System-Wert kann angegeben werden wie lange Ergebnisse einer Asynchron ausgeführten Service-Funktion maximal im System zum ersten Abruf bereit gehalten werden. Die Angabe erfolgt in Sekunden. Bei Vorgabe von 0 gibt es keine zeitliche Begrenzung für das Vorhalten.

Service-Zugang erlaubt Standard-Netzwerk Zugang

Mit dieser Option können Sie festlegen ob der Zugriff auf die WWSVC WEBWARE WEB Services über die Standardschnittstelle Ihrer WEBWARE-Instanz erlaubt ist. Bei Deaktivierung ist der Zugriff auf WWSVC-Funktionen über https://[Server-Name/WWSVC/.. nicht erlaubt und es wird eine 404er Fehlermeldung zurückgegeben.

Service-Zugang erlaubt Zusatz-Netzwerk Zugang

Mit dieser Option können Sie festlegen ob der Zugriff auf die WWSVC WEBWARE WEB Services über die Zusatzschnittstelle Ihrer WEBWARE-Instanz erlaubt ist. Bei Deaktivierung ist der Zugriff auf WWSVC-Funktionen über https://[Server-Name/WWSVC/.. nicht erlaubt und es wird eine 404er Fehlermeldung zurückgegeben. Diese Einstellung macht nur Sinn wenn Sie auch eine Zusatz-Schnitstelle im Bereich Konfiguration > System-Konfiguration > Netzwerk Anbindung > WEB Schnittstelle eingetragen haben.

Funktionsnamen aus HTTP-Befehlen ergänzen

Ist dieser Parameter aktiviert, so wird beim Zugriff auf Datenobjekte (Bspl: Artikel) ohne Angabe von Funktionen des Datenobjekts (Bspl: ARTIKEL.UPDATE, ARTIKEL.GET) der Funktionsname anhand des HTTP-Befehls ermittelt und zur Laufzeit für das Datenobjekt vorgegeben.

Details

Um Funktionen auszuführen haben Sie die Möglichkeit diese per normalen HTTP PUT/POST auszulösen. Dabei ist zu Beachten das der WWSVC-Server eingehende Funktionsnamen bei Zugriff auf eine Datenobjekt wie zum Beispiel ARTIKEL, BELEGE, (...) oder WORKFLOW je nach HTTP-Befehl auf festgelegte Funktionen routet. Dies geschieht jedoch nur wenn dieser Parameter hier aktiv ist.

Wird bei einem Funktionsaufruf direkt eine Funktion für ein Datenobjekt angegeben so entfällt das weitere Routen. (ARTIKEL.GET wurde angegeben so wird keine Funktion über HTTP gesetzt)

Beispiel (zur Vereinfachung im EXECURL Format):

Wenn Sie eine Funktion ausführen auf das Datenobjekt ARTIKEL und geben dabei keine weitere Ausführungs-Funktion an wie zum Beispiel (ARTIKEL.UPDATE, ARTIKEL.INSERT, ARTIKEL.DELETE ARTIKEL.GET), dann wird anhand des HTTP-Befehls die Ziel-Funktion automatisch angefügt.

HTTP	URL	Auführungsfunktion
PUT	/WWSVC/SVP/ARTIKEL	ARTIKEL.UPDATE
POST	/WWSVC/SVP/ARTIKEL	ARTIKEL.INSERT
GET	/WWSVC/SVP/ARTIKEL	ARTIKEL.GET
INSERT	/WWSVC/SVP/ARTIKEL	ARTIKEL.INSERT
DELETE	/WWSVC/SVP/ARTIKEL	ARTIKEL.DELETE

Beachten Sie das bei Ausführung mit dem GET-Befehl teilweise keine Übergabe von JSON-Parametern aus Browsern im Body-Bereich erfolgt.

Sie können also für den Zugriff auf die /WWSVC/EXEC... Funktionen folgende HTTP-Befehle verwenden:

- PUT entspricht Datenobjekt . UPDATE
- POST entspricht Datenobjekt . INSERT
- GET entspricht Datenobjekt . GET
- INSERT entspricht Datenobjekt . INSERT
- DELETE entspricht Datenobjekt . DELETE
- UPDATE entspricht Dtaenobjekt . UPDATE
- EXEC entspricht Datenobjekt . EXEC (Ausführen von Workflows usw.)
- OPTIONS Abfragen der API-JSON Struktur für das angegebene Objekt

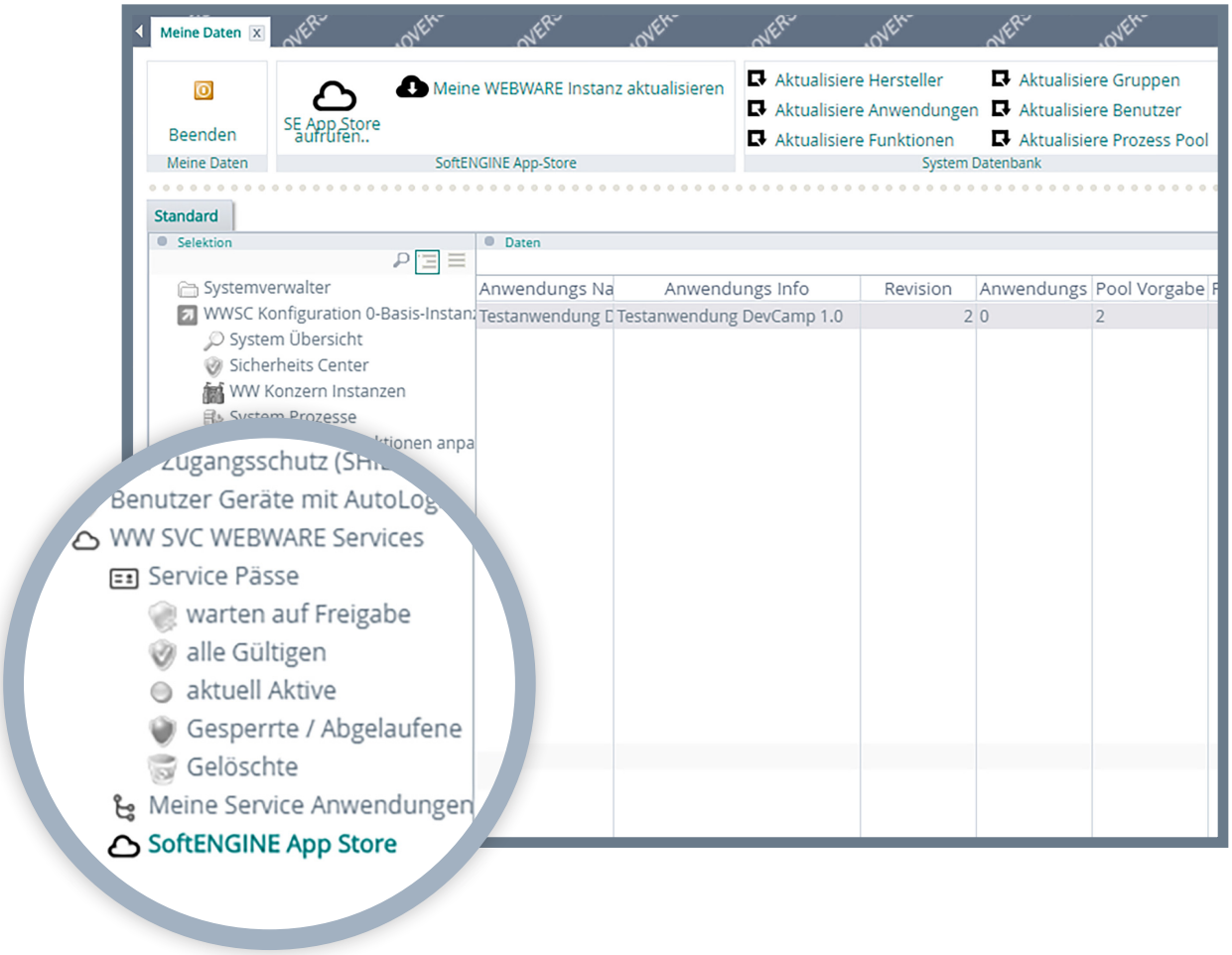
Setze RESULT_MAX_LINES bei 0 auf (System-Wert)

Wird bei einer Service-Funktion der Parameter RESULT_MAX_LINES nicht übergeben bzw. ist dieser 0 so wird diese System-Wert gesetzt um durch Fehlerhafte Abfragen keine Laufzeitprobleme auszulösen. Gerade bei großen Datenbeständen wird bei den Rückgabewerten sehr schnell mehrere 100 MB erreicht so dass die Übertragung zwischen den Systemen (WWS und Client) überdurchschnittlich lange dauert.

Werden mehr Sätze benötigt so kann dies der WWSVC-Schnittstelle durch HTTP-Header (WWSVC-ACCEPT-RESULT-MAX-LINES) bzw. JSON Attribut (GET_RESULT_MAX_LINES) mit angegeben werden.

Verwaltung von WWSVC-Elementen

Hier finden Sie Informationen welche Elemente im WW-SVC System vorkommen, und wie diese verwaltet werden. Sie finden im System-Cockpit in den Bereichen Administration/Konfiguration im Baum einen Eintrag "WW SVC WEBWARE Services".

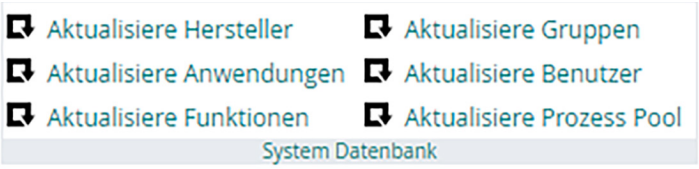


Unterhalb des Eintrages „WW SVC WEBWARE Services“ finden Sie die 3 Haupt-Objekte die für die WEBWARE-Services verwendet werden.

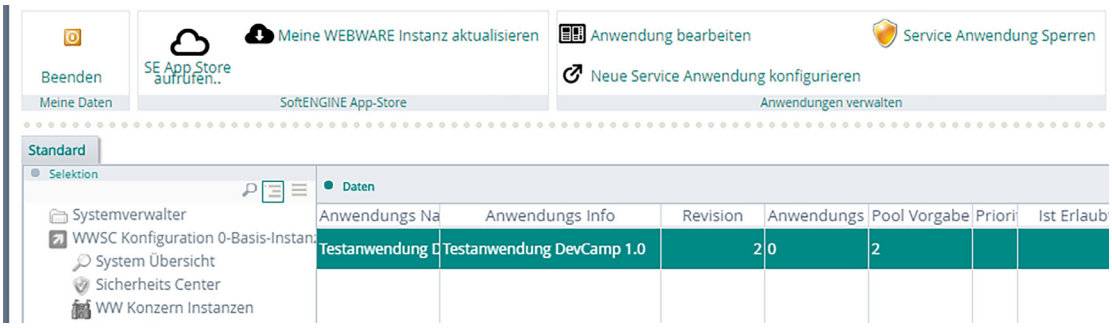
- SoftENGINE AppStore
- Meine Service-Anwendungen
- Service-Pässe

Um Ihrem WEBWARE-Server System neue Hersteller/Anwendungen bekannt zu machen, können Sie diese aus dem SoftENGINE AppStore laden („Meine WEBWARE Instanz aktualisieren“). Dabei werden neue Hersteller sowie deren Anwendungsbeschreibungen im Datenbankbereich (RAR-Server/DB) aktualisiert.

Nach der Aktualisierung werden dann die Information im RAR-Bereich in der Datenbank abgelegt. Mit Hilfe der "System-Datenbank Funktionen" können Sie dann ein sofortiges Aktualisieren der WW-Server System-Cockpit Datenbank auslösen.



Die Liste der erlaubten Anwendungen wird dabei im SoftENGINE AppStore verwaltet und kann von dort aus verändert werden.



Service Anwendungen (SecuredApp)

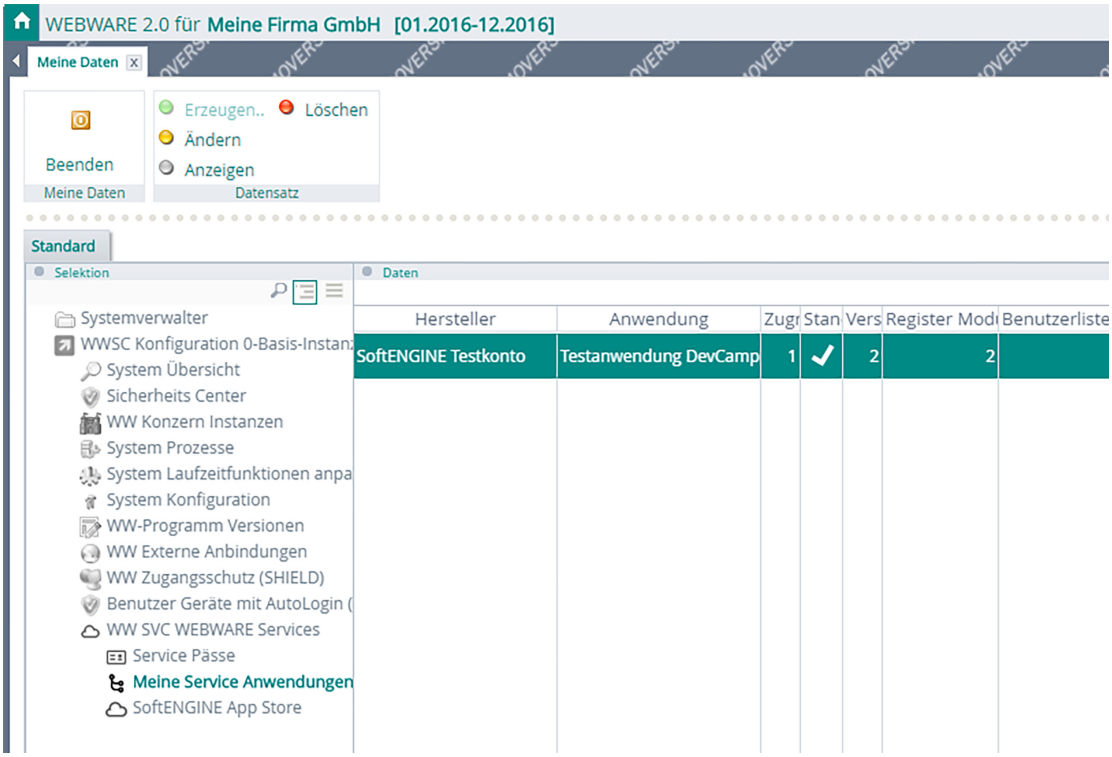
Wurde eine Hersteller-Anwendung freigegeben, so erscheint diese im Baum im Bereich "Service-Anwendung verwalten".

Eine Service Anwendung enthält Rahmenparameter und Zugriffsbeschränkungen mit denen Sie diese an Ihre Sicherheitsvorgaben anpassen können.

Unterhalb dieses Astes finden Sie für jeden freigegebenen Hersteller einen Eintrag, sowie unter jedem Hersteller dessen freigegebene Hersteller-Anwendungen.

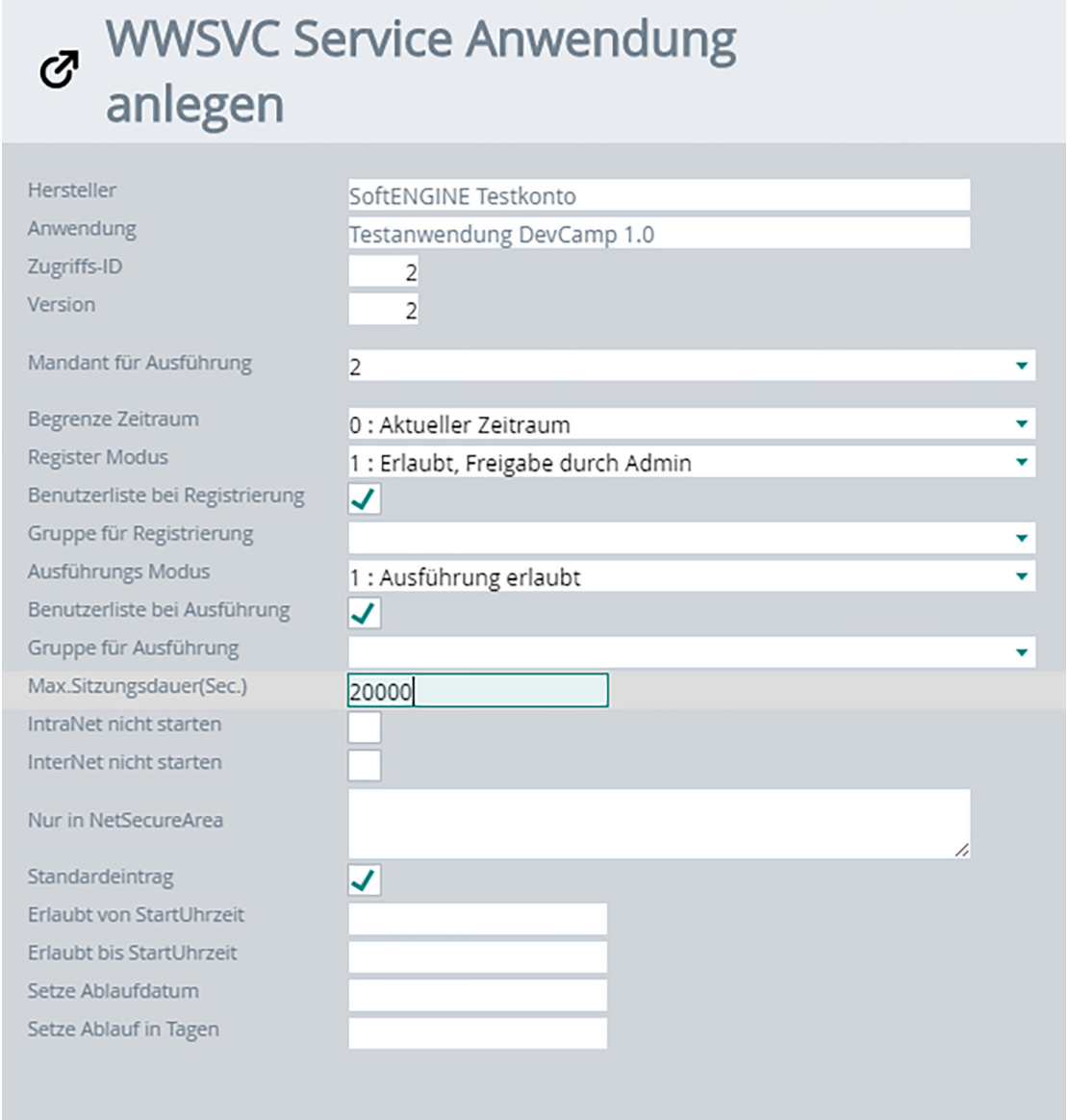
Unterhalb dieses Astes finden Sie für jeden freigegebenen Hersteller einen Eintrag, sowie unter jedem Hersteller dessen freigegebene Hersteller-Anwendungen.

Wird ein Hersteller unterhalb von „Service-Anwendungen verwalten“ ausgewählt, so erhalten Sie eine Liste der Secured-App‘s die Sie aus dem Bereich „Hersteller-Wendungen verwalten“ für Ihr WEBWARE-Server WWSVC-System freigegeben haben.



Aus dem Menü können Sie die Änderung von Vorgabeparameter für Ihre Secured-App vornehmen, oder auch Benutzer-Listen für die Zugriffs-Steuerung auf Ihr System unter Verwendung dieser Secured-App verwalten.

Folgende Parameter einer Service-Anwendung sind zu Bearbeiten



Zugriffs-ID

Die Zugriffs-ID ist ein eindeutiger Wer, welcher beim Speichern einer Service-Anwendung automatisch ermittelt wird. Bei der Registrierung eines Client's für diese Service-Anwendung muss die Zugriffs-ID angegeben werden, um eine Auswahl für Mandant und Belegzeitraum sowie die hier angegebenen Konfigurationswerte zu Erreichen.

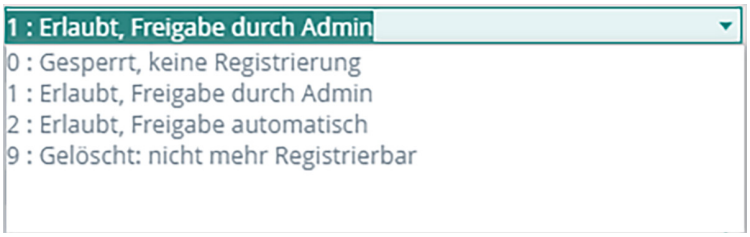
Mandant für Ausführung

Geben Sie hier den Mandanten vor der bei der Ausführung der Service-Funktion verwendet wird.

Begrenze Zeitraum

Wählen Sie hier den Belegzeitraum welcher für die Ausführung der Service-Funktion verwendet wird. Im Standard wird hier der Belegzeitraum 0 (Aktuelles Jahr) verwendet. Falls Sie Daten in abweichende Belegzeiträume schreiben wollen, müssen Sie hier den entsprechenden Zeitraum wählen.

Registrierungs-Modus



Sie können hier vorgeben ob diese Anwendung Registrierung Gesperrt oder Erlaubt ist. Bei erlaubten Anwendungen können Sie festlegen ob ein Administrator die Anwendung manuell freigeben muss.

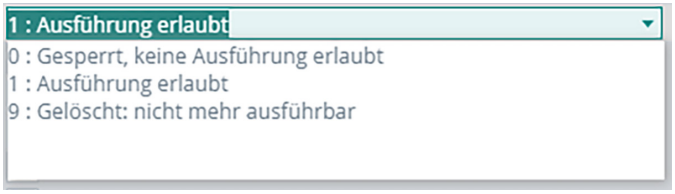
Benutzerliste bei Registrierung [verwenden]

Sie können hier angeben ob bei Registrierung für einen Service-Pass die Benutzerprüfung anhand einer Benutzer-Liste erfolgen muss. Ist dieser Parameter aktiviert, so wird bei der Registrierung der Benutzer-Name sowie das übergebene Benutzer-Passwort gegen die Benutzer-Liste die bei der Secured-App hinterlegt ist geprüft. Wird der Benutzer mit dem übergebenen Passwort nicht in der Benutzer-Liste gefunden, so wird der Service-Pass nicht erstellt und die Registrierung abgelehnt.

Benutzer-Gruppe für Registrierung [wer darf registrieren]

Hier können Sie aus der Benutzer-Gruppen Verwaltung der WWSVC eine Benutzergruppe für die Registrierung vorgeben. Bei Vorgabe und Aktivierung von "Benutzerliste bei Registrierung" können sich nur noch Benutzer anmelden die mit Benutzer-Kennung und Benutzer-Passwort in der Benutzer-Gruppe eingetragen sind.

Ausführungs Modus



Sie können hier vorgeben ob die Ausführung für registrierte Service Anwendungen gesperrt, oder erlaubt ist. Ist die Ausführung gesperrt, so können auch gültige Service-Pässe nicht auf diese Anwendung zugreifen.

Benutzerliste bei Ausführung [verwenden]

Sie können hier angeben ob bei Verwendung eines Service-Passes bzw. die Benutzung einer Secured-App zuvor eine Benutzer-Anmeldung (Authentifizierung) mit Benutzer und Passwort gegen die bei der Secured-App hinterlegten Benutzer-Liste erfolgen muss. Ist dieser System-Wert aktiviert so erhält der Aufrufer die Fehlermeldung "401-Authorization required" (Benutzer-Anmeldung erforderlich) wenn er ohne erfolgreiche Anmeldung seinen Service-Pass verwendet.

Die Anmeldung muss, bei Aktivierung dieses System-Wertes, über die WWSVC/WWSERVICE/CONNECT-Funktion erfolgen. Dabei wird Service-Pass + Benutzer-Name + Benutzer-Passwort übergeben, und der Aufrufer erhält bei Erfolg einen [zeitlich beschränkten] Sitzungs-Schlüssel übergeben, der bei jedem weiteren Zugriff übergeben werden muss.

Benutzer-Gruppe für Ausführung[wer darf ausführen]

Hier können Sie aus der Benutzer-Gruppen Verwaltung der WWSVC eine Benutzergruppe für die Ausführung vorgeben. Bei Vorgabe und Aktivierung von "Benutzerliste bei Ausführung" können nur noch Benutzer Funktionen für diese Service-Anwendung ausführen die mit Benutzer-Kennung und Benutzer-Passwort in der Benutzer-Gruppe eingetragen sind.

Benutzerliste maximale Sitzungsdauer [in Sekunden]

Wird eine Benutzerliste für den Zugriff verwendet, so erhält der Client bei erfolgreicher Anmeldung (Authentifizierung) einen Sitzungs-Schlüssel. Hier können Sie nun angeben wie lange dieser Sitzungsschlüssel maximal gültig sein soll.

0 : Keine zeitliche Begrenzung, nach einmaliger Anmeldung bleibt der Sitzungsschlüssel immer gültig, welcher bei jedem Zugriff zusätzlich angegeben werden muss.

>0: Zeit in Sekunden bis zum Ablauf des Sitzungsschlüssels. Der Sitzungsschlüssel muss immer mit angegeben werden. Ist der Sitzungsschüssel nicht

mehr gültig, so erhält der Client die Fehlermeldung "401-Authorization required" Anmeldung erforderlich als Fehler-Code

IntraNet / InterNet nicht starten

Sie können mit diesen 2 Parametern festlegen ob der Zugriff aus Inter-/Intranet verboten ist.

Nur in NetSecureArea

Hier können Sie einen IP-Adress-Bereich vorgeben für den sicheren Betrieb. Wenn hier ein Wert angegeben wird, so können nur Client's aus dem erlaubten Bereich diese Anwendung starten.

Standardeintrag

Hier können Sie die Version der Service-Anwendung auf die aktuelle Beschränken, bzw. den Zugriff auch von älteren Versionen erlauben (Wird später noch genauer implementiert)

Erlaubt von / bis Uhrzeit

Hier können Sie den Zugriff auf einen Zeitbereich am Tag einschränken.

Begrenze auf Mandant/Zeitraum

Hier können Sie den Mandant der für diese Service-Anwendung verwendet wird vorgeben, sowie den Belegzeitraum.

Setze Ablaufdatum

Hier können Sie eine Datum setzen, bis zu diesem der Zugriff von gültigen Service-Pässen für diese Anwendung erlaubt ist.

Setze Ablauf in Tagen

Hier können Sie ein Ablaufterm in Tagen setzen ab wann der Zugriff von gültigen Service-Pässen für diese Anwendung erlaubt ist.

Verwalten von Benutzer-Listen

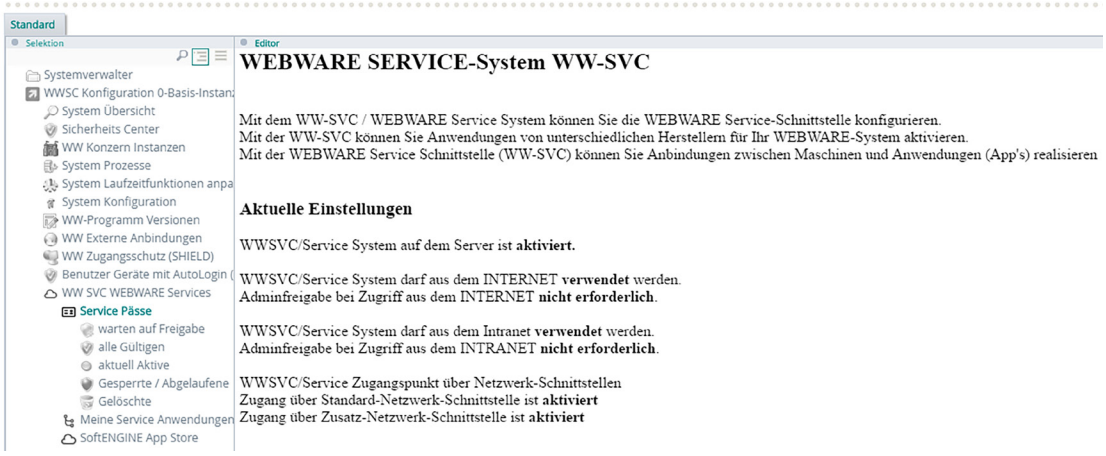
Mit dem Menü-Befehl "Benutzerlisten verwalten" können Sie die Benutzerverwaltung für WWSVC aufrufen. Die Datenhaltung erfolgt dabei im RAR-Datenbankbereich. Sie haben hier die Möglichkeit Benutzergruppen anzulegen, sowie Benutzer mit Passwort dort je Gruppe zu verwalten.
Wenn Sie in Ihrer "Service Anwendung", Vorgaben für die Verwendung von Benutzer-Listen gemacht haben, so ist der Zugriff, also Registrierung und/oder Verwendung der "Service Anwendung" für einen Service-Pass nur möglich wenn der Client den korrekten Benutzer-Name und Benutzer-Passwort übergibt.

Je nachdem ob Sie eine Benutzergruppe für Registrierung und/oder Ausführung bei einer Service-Anwendung hinterlegt haben, wird im Menü der entsprechende Befehl mit angezeigt. Damit können Sie direkt die Benutzer-Listen verwalten.

Verwalten von Benutzer-Listen im Bereich Administration

Wenn Sie sich im Bereich Administration anmelden, können Sie dort im Bereich Zugangs-Verwaltung unter WW-SVC-Services BenutzerListen für die freigegebenen Service-Anwendungen die zugehörigen Benutzerlisten verwalten.

Service Pässe

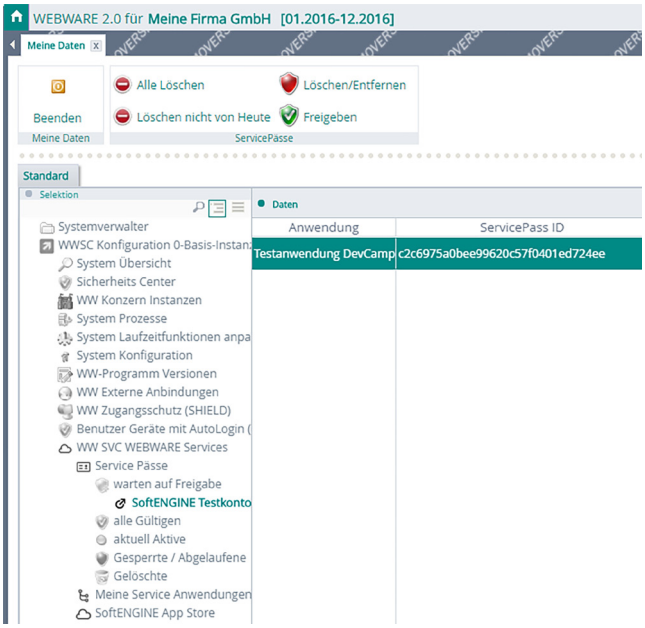


Will ein Client auf Ihr WEBWARE System zugreifen, so benötigt er einen gültigen Service-Pass. Hierzu muss er wie weiter oben beschrieben sich bei Ihrem WEBWARE System registrieren. Je nach Vorgaben ist eine automatische Registrierung sowie eine manuelle Freigabe vom System-Administrator möglich. Ist der Zugriff erlaubt, so erhält der Client einen Service-Pass mit dem er von nun an auf Ihr WEBWARE System und der damit verbundenen Secured Service Anwendung zugreifen kann.

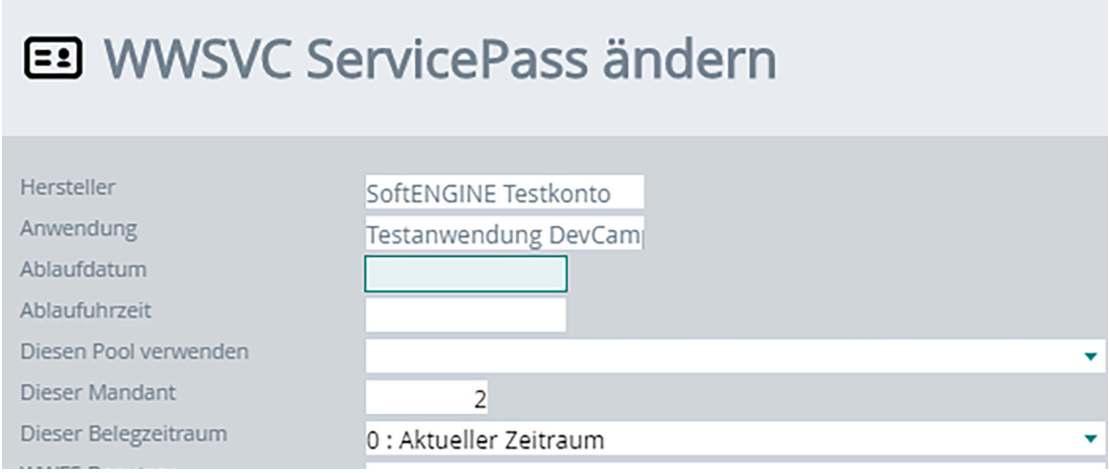
Die Service Pässe werden nach Ihrem Status im Baum dargestellt.

Warten auf Freigabe

Neue Service-Pässe die nicht automatisch freigegeben werden sollen erscheinen in diesem Bereich. Der Administrator hat die Möglichkeit diese zu Löschen (Alle, Löschen älter wie heute, Einzel-Löschung), sowie den ServicePass freizugeben, also für den Zugriff zu aktivieren.



Wird ein Service-Pass angeklickt, so können Sie folgende Parameter verwalten:



Abaufdatum

Datum bis zu welchem der Service-Pass gültig ist

Ablaufuhrzeit

Ist das Ablaufdatum gesetzt, so wird der Eintrag ab der entsprechenden Uhrzeit gesperrt.

Diesen Pool verwenden

Hier können Sie einen Prozess-Pool angeben, in dem eingehende Service-Funktionen für diesen Service-Pass verarbeitet werden sollen.

Dieser Mandant

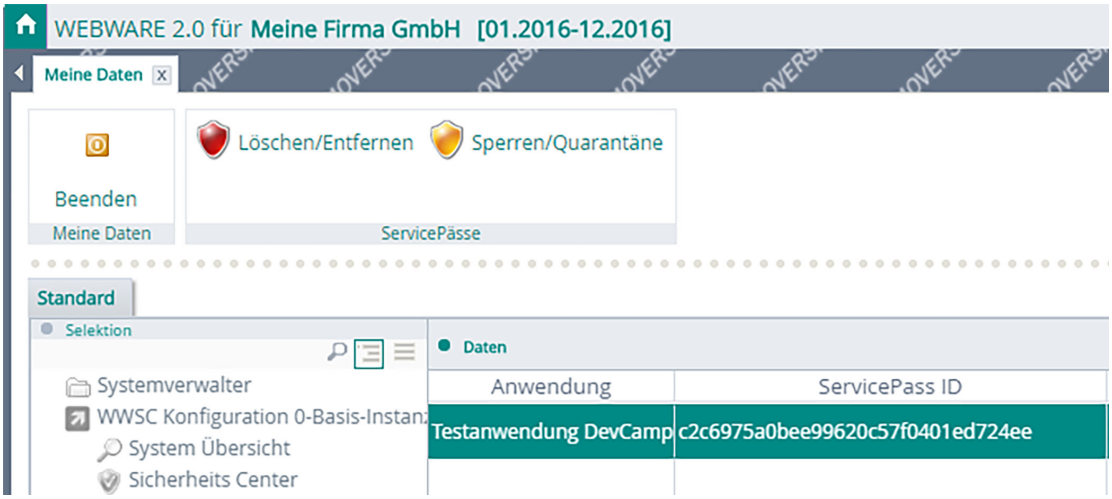
Hier wird der Mandant ihres WEBWARE Sytemes vorgeben auf welchen dieser Service-Pass zugreifen darf/kann.

Dieser Belegzeitraum

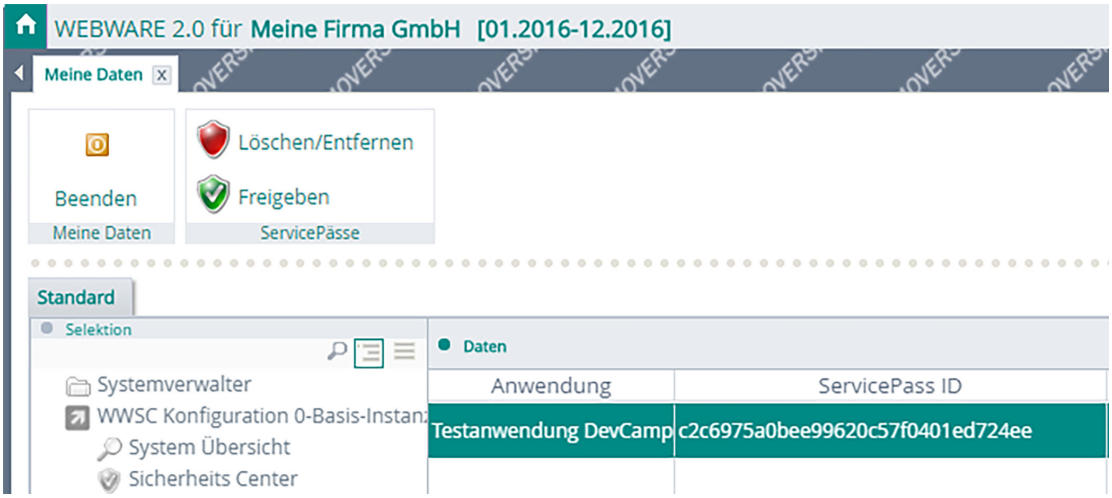
Hier wird der Belegzeitraum vorgegeben auf welchen der Service-Pass zugreifen kann.

Alle Gültigen ServicePässe

Hier finden Sie eine Übersicht über alle gültigen ServicePässe, bzw. unterhalb je Hersteller diese sortiert. Sie können hier einzelne ServicePässe Sperren, bzw. komplett Löschen.



Gesperrte/Abgelaufene Service-Pässe



Hier können Sie gesperrte bzw. abgelaufene ServicePässe verwalten und bei Bedarf diese wieder Freigeben bzw. komplett Löschen.

Beschreibung der Servicefunktionen

Referenz der Servicefunktionen

In diesem Abschnitt finden Sie eine Referenz der verfügbaren Servicefunktionen. Bitte beachten: Während der BETA-Phase sind jederzeit Änderungen bei Ein- und Ausgabeformaten sowie der Namen der Funktionen und Ressourcen möglich.

Allgemeines

Alle Funktionen liefern bei Erfolg den HTTP-Status „200 OK“. Im Fehlerfall werden in der COMRESULT-Struktur detaillierte Fehlerinformationen mitgeliefert. Das Namensschema ist stets RESOURCE.FUNKTION. Eine RESOURCE ist dabei eine Datenbanktabelle, ein Workflowskript oder eine Auswertung. FUNKTION ist der Name der Funktion, die auf der Ressource ausgeführt wird. Alle Ressourcen besitzen die parameterlose Funktion OPTIONS, die eine Liste der für die Ressource implementierten Funktionen liefert.

ARTIKEL.GET

Holt eine Liste von Artikeldatensätzen.

Parameter (optional):

- VON_ARTNR
- BIS_ARTNR — schränkt die Liste auf den angegebenen Artikelnummernbereich ein
- VON_WGR
- BIS_WGR — schränkt die Liste auf den angegebenen Warengruppenbereich ein
- VON_KATALOG
- BIS_KATALOG — schränkt die Liste auf den angegebenen Katalognummernbereich ein

Verwendbare Standardparameter (optinoal):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=ART_1_25, ART_51_60)
- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze
- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets
- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.
- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (ART_36_5='WGR01' # ART_36_5='WGR56')

Rückgabe (normal):

ARTIKELLISTE

```
{
ARTIKEL
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

ARTIKEL.PUT

Aktualisiert einen vorhandenen Artikel

Parameter (obligatorisch):

- ARTNR Artikelnummer

Parameter (optional)

- ART_... zu setzende Felder
- OHNE_STAMMKALK 1 = keine Stammdatenkalkulation durchführen

Rückgabe:

Keine

ARTIKEL.INSERT

Legt einen neuen Artikel an.

Parameter (optional):

- ART_... zu setzende Felder
- ARTNR Vorgabe für die Artikelnummer
- ERFASSUNGSGRUPPE zu verwendende Artikelerfassungsgruppe
- NUR_TESTEN 1 = Vorgang nur testen, es wird kein Satz angelegt

Rückgabe:

{
ARTNR: <die neu vergebene Artikelnummer>
}

Hinweis: wird der Parameter ARTNR nicht mitgeliefert, wird eine neue Artikelnummer automatisch erzeugt.

ARTIKEL.DELETE

Löscht einen Artikeldatensatz.

Parameter (obligatorisch):

- ARTNR die Artikelnummer des zu löschenden Artikels

Rückgabe:

Keine

Hinweis: es wird eine Überprüfung durchgeführt, ob der Artikel gelöscht werden kann. Artikel mit Umsatz können nicht gelöscht werden.

ADRESSE.GET

Holt eine Liste von Adressdatensätzen.

Parameter (optional):

- VON_ADRNR
- BIS_ADRNR — schränkt die Liste auf den angegebenen Artikelnummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=ADR_2_8, ADR_10_10)
- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze
- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets
- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.
- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (ADR_2_8 < 70000 # ADR_10_10='abcdef')

Rückgabe:

ADRESSLISTE

```
{
        ADRESSE
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

ADRESSE.PUT

Aktualisiert einen bestehenden Adressdatensatz

Parameter (obligatorisch):

- ADRNR Adressnummer

Parameter (optional):

- ADR_... zu setzende Felder
- OHNE_STAMMKALK 1=keine Stammdatenkalkutlation durchführen

Rückgabe:

Keine

ADRESSE.INSERT

Erstellt einen neuen Adressdatensatz.

Parameter (obligatorisch):

- TYP — Typ der Adresse(KUNDE, LIEFERANT, ERSTKONTAKT)

Parameter (optional):

- ADR_... — zu setzende Felder
- ADRNR — Vorgabe für die Artikelnummer
- ADRART — Vorgabe Adressart
- ADRKREIS — Vorgabe Adresskreis
- LAND — Vorgabe Ländereinstellungen
- NUR_TESTEN — 1 = Vorgang nur testen, es wird kein Satz angelegt

Rückgabe:

{
ADRNR: <Adressnummer des neu erstellten Datensatzes>
}

Hinweis: wird der Parameter ARTNR nicht mitgeliefert, wird eine neue Artikelnummer automatisch erzeugt.

ADRESSE.DELETE

Löscht einen Adressdatensatz.

Parameter (obligatorisch):

- ADRNR Adressnummer

Rückgabe:

Keine

Hinweis:
Es wird eine Prüfung durchgeführt, ob der Adressdatensatz gelöscht werden kann.

BELEG.GET

Holt eine Liste von Belegkopfdatensätzen

Parameter (obligatorisch):

- VON_ADR
- BIS_ADR — schränkt die Liste auf den angegebenen Adressnummernbereich ein

- VON_BELNDX
- BIS_BELNDX — schränkt die Liste auf den angegebenen Belegindexbereich ein

- VON_JAHR
- BIS_JAHR — schränkt die Liste auf das angegebene Geschäftsjahr ein (0 - 9)

- BELART — schränkt die Liste auf die angegebene Belegart ein

- VON_BELNR
- BIS_BELNR — schränkt die Liste auf den angegebenen Beleg nummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=BEL_0_1, BEL_1_1, BEL_2_1, BEL_3_8)
- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze
- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets
- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.
- FREISELEKT — übergibt einen freien Selektionsausdruck Bsp.: (BEL_19_10>='01.01.2016' & BEL_19_10='31.01.2016')

Rückgabe:

BELEGLISTE

```
{
        BELEG
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

BELEG.PUT

Aktualisiert einen vorhandenen Beleg

Parameter (obligatorisch):

- BELNDX — der Index des Belegs
- Parameter (optional)
- BEL_... — zu setzende Felder
- BERECHNE_SUMMEN — 1=Belegsummern werden neu berechnet
- KEINE_VERBUCHUNG — 1= es wird keine Verbuchung durchgeführt
- KALKNDX — abweichender Index der verwendeten Stammdatenkalkulation

Rückgabe:

Keine

BELEG.INSERT

Erstellt einen Warenwirtschaftsbeleg

Parameter (obligatorisch):

- BEL_2_1 Belegart
- BEL_11_8 Adressnummer

Parameter (optional):

- BEL_... weitere Belegkopffelder

Belegpositionsdaten können als JSON-Substruktur wie folgt übergeben werden:

POSDATEN

```
{
        POSITION:
        [
                {
                        POS_18_25           Artikelnummer (obligatorisch)
                        POS_164_8           Menge (obligatorisch)
                        POS_...             weitere Positionsfelder (optional)
                },
                ...
        ]
}
```

Rückgabe:

```
{
BELNDX: <Index des neu angelegten Belegs>
POS_ANGELEGT: <Anzahl angelegter Positionen>
}
```

BELEG.DELETE

Löscht einen Belegkopfsatz inklusive seiner Positionssätze.

Parameter (obligatorisch):

- BELNDX der Primaärschlüssel des zu löschenden Belegs

Parameter (optional):

- KEINE_ARCHIVIERUNG 1 = es wird keine Archivierung durchgeführt

Rückgabe:

Keine

BELPOS.GET

Holt die Positionen zu einem Beleg

Parameter (obligatorisch):

- BELNDX — vollständiger Belegindex

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=POS_17_1, POS_18_25)
- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze
- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets

Rückgabe:

POSITIONSLISTE

```
{
        POSITION
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

BELPOS.PUT

Aktualisiert einen vorhanden Belegpositionssatz

Parameter (obligatorisch):

- SNR Die Satznummer des zu aktualisierenden Satzes

Parameter (optional):

- MIT_VERBUCHUNG 1=Der Satz wird für die Statistik verbucht
- MIT_BERECHNUNG 1=Die Formel wird ausgeführt

Rückgabe:

Keine

BELPOS.INSERT

Fügt eine neue Belegposition ein.

Parameter (obligatorisch):

- BELNDX Primärschlüssel des Belegs
- POS_18_25 Artikelnummer
- POS_164_8 Menge

Parameter (optional):

- LANGTEXT_AUFLOESEN 1=Langtext wird aufgelöst
- EINFUEGE_SNR Satznummer des Vorgängers

Rückgabe:

{
SNR: <Satznummer des eingefügten Satzes>
}

IDB<IDBID>.GET

Holt eine Liste von Datensätzen aus der IDB mit der ID <IDBID>.

Parameter (optional):

- VON_PK
- BIS_PK — schränkt die Liste auf den angegebenen Primärschlüsselbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=IDB_2_8, IDB_10_10)
- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze
- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets
- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.
- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (IDB_0_3='AAA' # IDB_0_3='ZZZ')

Rückgabe:

IDB<IDBID>LISTE

```
{
        IDB<IDBID>
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

IDB<IDBID>.PUT

Aktualisiert einen vorhandenen IDB-Satz.

Parameter (obligatorisch):

- PK Primärschlüssel

Parameter (optional):

- IDB_... zu setzende Felder
- OHNE_STAMMKALK 1=keine Stammdatenkalkulation durchführen

Rückhabe:

Keine

IDB<IDBID>.INSERT

Erstellt einen neuen IDB-Satz.

Parameter (optional):

- PK — Vorgabe für Primärschlüssel
- OHNE_STAMMKALK — 1 = Keine Stammdatenkalkulation durchführen
- NUR_TESTEN — 1 = Vorgang nur testen, es wird kein Satz angelegt.

Rückgabe:

{
PK: <Primärschlüssel des neu erstellen Datensatzes>
}

Hinweis: wird der Parameter PK nicht mitgeliefert, kann ein neuer Primärschlüssel nur dann automatisch erzeugt werden, sofern dies in den IDB-Einstellungen hinterlegt ist.

IDB<IDBID>.DELETE

Löscht einen IDB-Satz

Parameter (obligatorisch):

- PK der Primärschlüssel des zu löschenden Datensatzes

Rückgabe:

Keine

PROJEKT.GET

Holt eine Liste von Projektdatensätzen

Parameter (optional):

- VON_PRJNR
- BIS_PORJNR — schränkt die Liste auf den angegebenen Projektnummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=PRJ_1_8, PRJ_9_8)
- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze
- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets
- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.
- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (PRJ_9_8 < 20000 # PRJ_9_8 > 21000)

Rückgabe:

PROJEKTLISTE

```
{
        PROJEKT
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

PROJEKT.PUT

Aktualisiert einen vorhandenen Projektsatz

Parameter (obligatorisch):

- PRJNR Projektnummer

Parameter (optional):

- PRJ_... zu setzende Felder
- OHNE_STAMMKALK 1 = keine Stammdatenkalklulation durchführen

Rückgabe:

Keine

PROJEKT.INSERT

Erstellt einen neuen Projektsatz

Parameter (optional):

- PRJNR Vorgabe für Projektnummer
- OHNE_STAMMKALK 1 = Keine Stammdatenkalkulation durchführen
- NUR_TESTEN 1 = Vorgang nur testen, es wird kein Satz angelegt.

Rückgabe:

{
PRJNR: <Projektnummer des neu erstellen Datensatzes>
}

Hinweis: Wird der Parameter PRJNR nicht mitgeliefert, wird eine neue Projektnummer automatisch erzeugt.

PROJEKT.DELETE

Löscht einen Projektdatensatz.

Parameter (obligatorisch):

- PRJNR Projektnummer

Rückgabe:

Keine

Hinweis:
Es wird eine Prüfung durchgeführt, ob der Projektdatensatz gelöscht werden kann.

SERIENNUMMER.GET

Holt eine Liste von Seriennummern

Parameter (optional):

- VON_SERNR
- BIS_SERNR — schränkt die Liste auf den angegebenen Seriennummernbereich ein

- VON_ARTNR
- BIS_ARTNR — schränkt die Liste auf den angegebenen Artikelnummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=SER_1_25, SER_26_25)

- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze

- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets

- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.

- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (SER_26_25='Artikel123' # SER_26_25='Artikel124')

Rückgabe:

SERIENNUMMERNLISTE

```
{
        SERIENNUMMER
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

SERIENNUMMER.PUT

Aktualisiert einen vorhandenen Seriennummernsatz

Parameter (obligatorisch):

- SERNR Seriennummer

Parameter (optional):

- SER_... zu setzende Felder
- OHNE_STAMMKALK 1 = keine Stammdatenkalkulation durchführen

Rückgabe:

Keine

SERIENNUMMER.INSERT

Erstellt einen neuen Seriennummernsatz

Parameter (obligatorisch):

- SERNR Seriennummer

Parameter (optional):

- SER_... zu setzende Felder
- OHNE_STAMMKALK 1 = Keine Stammdatenkalkulation durchführen
- NUR_TESTEN 1 = Vorgang nur testen, es wird kein Satz angelegt.

Rückgabe:

{

SERNR: <Primärschlüssel des neu erstellen Datensatzes>

}

SERIENNUMMER.DELETE

Löscht einen Seriennummernsatz

Parameter (obligatorisch):

- SERNR Seriennummer

Rückgabe:

Keine

CHARGE.GET

Holt eine Liste von Chargen

Parameter (optional):

- VON_CHANR
- BIS_CHANR — schränkt die Liste auf den angegebenen Chargennummernbereich ein

- VON_ARTNR
- BIS_ARTNR — schränkt die Liste auf den angegebenen Artikelnummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=CHA_1_25, CHA_26_25)

- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze

- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets

- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.

- FREISELEKT — übergibt einen freien Selektionsausdruck Bsp.: (CHA_26_25='Artikel123' # CHA_26_25='Artikel124')

Rückgabe:

CHARGENLISTE

```
{
        CHARGE
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

CHARGE.PUT

Aktualisiert einen vorhandenen Chargensatz

Parameter (obligatorisch):

- CHANR Chargennummer

Parameter (optional):

- CHA_... zu setzende Felder
- OHNE_STAMMKALK 1 = keine Stammdatenkalkulationen durchführen

Rückgabe:

Keine

CHARGE.INSERT

Erstellt einen neuen Chargensatz

Parameter (obligatorisch):

- CHANR Chargennnummer

Parameter (optional):

- OHNE_STAMMKALK 1 = Keine Stammdatenkalkulation durchführen
- NUR_TESTEN 1 = Vorgang nur testen, es wird kein Satz angelegt.

Rückgabe:

{

CHANR: <Primärschlüssel des neu erstellen Datensatzes>

}

CHARGE.DELETE

Löscht einen Chargensatz

Parameter (obligatorisch):

- CHANR Chargennummer

Rückgabe:

Keine

ADRESSARTIKEL.GET

Holt eine Liste von Artikeldatensätzen.

Parameter (optional):

- VON_ADRNR
- BIS_ADRNR — schränkt die Liste auf den angegebenen Adressnummernbereich ein

- VON_ARTNR
- BIS_ARTNR — schränkt die Liste auf den angegebenen Artikelnummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=ADA_1_8, ADA_9_25)

- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze

- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets

- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.

- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (ADA_34_25='123' # ADA_59_8 > 10)

Rückgabe (normal):

ADRESSARTIKELLISTE

{

ADRESSARTIKEL

[

{

SNR: <Satznummer>

Felder gemäß Feldliste oder alle Felder

},

...

]

ANZAHL: <Anzahl gelieferter Datensätze>

}

Rückgabe (NUR_ANZAHL):

{

ANZAHL: <Anzahl der Datensätze>

}

Rückgabe (NUR_GROESSE):

{

BYTES: <Größe in Bytes>

}

ADRESSARTIKEL.PUT

Aktualisiert einen vorhandenen Adressartikel

Parameter (obligatorisch):

- ADRNR Adressnummer
- ARTNR Artikelnummer

Parameter (optional):

- ADA_... zu setzende Felder
- OHNE_STAMMKALK 1 = keien Stammdatenkalkulation durchführen

Rückgabe:

Keine

ADRESSARTIKEL.INSERT

Legt einen neuen Adressartikel an.

Parameter (obligatorisch):

- ADRNR Adressnummer
- ARTNR Artikelnummer
- Parameter (optional):
- ADA_... zu setzende Felder

Rückgabe:

{

ADANDX: <Primärschlüssel des neu angelegten Datensatzes>

}

ADRESSARTIKEL.DELETE

Löscht einen Adressartikelsatz.

Parameter (obligatorisch):

- ADRNR Adressnummer
- ARTNR Artieklnummer

Rückgabe:

Keine

Hinweis: es wird eine Überprüfung durchgeführt, ob der Datensatz gelöscht werden kann. Abhängig von Zugriffsrechten kann es sein, dass ein Datensatz nicht gelöscht werden kann.

LIEFERADRESSE.GET

Holt eine Liste von Lieferadressdatensätzen.

Parameter (optional):

- VON_ADRNR
- BIS_ADRNR — schränkt die Liste auf den angegebenen Adressnummernbereich ein

- VON_LFANR
- BIS_LFANR — schränkt die Liste auf den angegebenen Lieferadressnummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=LFA_2_8, LFA_10_8)

- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze

- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets

- FREISELEKT — übergibt einen freien Selektionsausdruck.
  Bsp.: (LFA_110_10 >= 66000 # LFA_110_10 <= 66999)

Rückgabe:

LIEFERADRESSLISTE

```
{
        LIEFERADRESSE
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

LIEFERADRESSE.PUT

Aktualisiert einen bestehenden Lieferadressdatensatz

Parameter (obligatorisch):

- LFANR Lieferadressnummer

Parameter (optional):

- LFA_... zu setzende Felder
- OHNE_STAMMKALK 1 = keine Stammdatenkalkulation durchführen

Rückgabe:

Keine

LIEFERADRESSE.INSERT

Erstellt einen neuen Lieferadressdatensatz.

Parameter (optional):

- LFA_... zu setzende Felder

- LFANSPRECHPARTNER.GET
- Holt eine Liste von Ansprechpartnerdatensätzen.
- Parameter (optional): Vorgabe für Lieferadressnummer

- OHNE_STAMMKALK 1 = Keine Stammdatenkalkulation durchführen

Rückgabe:

{
LFANR: <Lieferadressnummer des neu erstellen Datensatzes>
}

Hinweis: wird der Parameter LFANR nicht mitgeliefert, wird eine neue Lieferadressnummer automatisch erzeugt.

LIEFERADRESSE.DELETE

Löscht einen Lieferadressdatensatz.

Parameter (obligatorisch):

- LFANR Lieferadressnummer

Rückgabe:

Keine

Hinweis:
Es wird eine Prüfung durchgeführt, ob der Lieferadressdatensatz gelöscht werden kann.

ANSPRECHPARTNER.GET

Holt eine Liste von Ansprechpartnerdatensätzen.

Parameter (optional):

- VON_ADRNR
- BIS_ADRNR — schränkt die Liste auf den angegebenen Adressnummernbereich ein

- VON_ANPNR
- BIS_ANPNR — schränkt die Liste auf den angegebenen Ansprechpartnernummernbereich ein

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=ANP_1_8, ANP_9_8)

- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze

- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets

- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.

- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (ANP_1120_20='+49170123456789')

Rückgabe:

ANSPRECHPARTNERLISTE

```
{
        ANSPRECHPARTNER
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

ANSPRECHPARTNER.PUT

Aktualisiert einen bestehenden Ansprechpartnerdatensatz

Parameter (obligatorisch):

- ADRNR Adressnummer
- ANPNR Ansprechpartnernummer

Parameter (optional):

- ANP_... zu setzende Felder
- OHNE_STAMMKALK 1 = keine Stammdatenkalkulation durchführen

Rückgabe:

Keine

ANSPRECHPARTNER.INSERT

Erstellt einen neuen Ansprechpartnerdatensatz.

Parameter (obligatorisch):

- ADRNR Adressnummer

Parameter (optional):

- ANP_... zu setzende Felder
- ANPNR Vorgabe für Ansprechpartnernummer
- OHNE_STAMMKALK 1 = Keine Stammdatenkalkulation durchführen

Rückgabe:

{
ANPNR: <Adressnummer des neu erstellen Datensatzes>
}

Hinweis: wird der Parameter ANPNR nicht mitgeliefert, wird eine neue Ansprechpartnernummer automatisch erzeugt.

ANSPRECHPARTNER.DELETE

Löscht einen Ansprechpartnerdatensatz.

Parameter (obligatorisch):

- ADRNR Adressnummer
- ANPNR Ansprechpartnernummer

Rückgabe:

Keine

Hinweis:
Es wird eine Prüfung durchgeführt, ob der Ansprechpartnerdatensatz gelöscht werden kann.

TERMIN.GET

Holt eine Liste von Terminen.

Parameter (optional):

- VON_PERSNR
- BIS_PERSNR — schränkt die Liste auf den angegebenen Personalnummernbereich ein

- VON_DATUM
- BIS_DATUM — schränkt die Liste auf den angegebenen Datumsbereich ein.

Verwendbare Standardparameter (optional):

- FELDER — übergibt eine Komma-getrennte Liste der gewünschten Datenfelder (z.B. FELDER=ANP_1_8, ANP_9_8)

- NUR_ANZAHL — 1=Liefert als Antwortpaket nur die Anzahl der gefundenen Datensätze

- NUR_GROESSE — 1=Liefert als Antwortpaket nur die Größe in Bytes des tatsächlichen Antwortpakets

- SUCHE_VOLLTEXT — übergibt einen Volltext-Suchbegriff. Die Volltextsuche wird als erster Schritt durchgeführt, danach werden weitere Selektionsparameter angewandt.

- FREISELEKT — übergibt einen freien Selektionsausdruck. Bsp.: (TRM_51_8= 10000)

Rückgabe:

TERMINLISTE

```
{
        TERMIN
        [
                {
                        SNR: <Satznummer>
                        Felder gemäß Feldliste oder alle Felder
                },
                ...
        ]
        ANZAHL: <Anzahl gelieferter Datensätze>
}
```

Rückgabe (NUR_ANZAHL):

```
        {
ANZAHL: <Anzahl der Datensätze>
        }
```

Rückgabe (NUR_GROESSE):

```
        {
BYTES: <Größe in Bytes>
}
```

TERMIN.PUT

Aktualisiert einen bestehenden Termin.

Parameter (obligatorisch):

- PK Primärschlüssel

Parameter (optional):

- TRM_... zu setzende Felder
- OHNE_STAMMKALK 1 = keine Stammdatenkalkulation durchführen

Rückgabe:

Keine

TERMIN.INSERT

Erstellt einen neuen Termin.

Parameter (obligatorisch):

- PERSNR — Personalnummer
- VON_DATUM — Startdatum
- TITEL — Titel des Termins

Parameter (optional):

- TRM_... — zu setzende Felder
- BIS_DATUM — Enddatum
- VON_ZEIT — Startzeit
- BIS_ZEIT — Endezeit
- TEXT — Terminbeschreibung
- OHNE_STAMMKALK — 1 = Keine Stammdatenkalkulation durchführen

Rückgabe:

{
PK: <Primärschlüssel des neu erstellen Datensatzes>
}

TERMIN.DELETE

Löscht einen Termin.

Parameter (obligatorisch):

- PK Primärschlüssel

Rückgabe:

Keine

Hinweis:
Es wird eine Prüfung durchgeführt, ob der Termin gelöscht werden kann.

GET_RELATION.EXEC

Führt eine GET_RELATION aus.

Parameter (obligatorisch):

- NR Die Nummer der GET_RELATION

Parameter (optional):

- P1..Pn weitere Parameter der GET_RELATION

Anmerkung: Die Parameter P1 bis Pn können in beliebiger Reihenfolge angegeben werden. Die Nummer entspricht dabei der Position des Parameters in dem vergleichbaren Aufruf der GET_RELATION in einem Eventskript. Soll ein Parameter leer bleiben, muss er mit leerem Inhalt übergeben werden.

Rückgabe:

{

GET_RESULT: <Rückgabewert der GET_RELATION>

}

PUT_RELATION.EXEC

Führt eine PUT_RELATION aus.

Parameter (obligatorisch):

- NR Die Nummer der PUT_RELATION

Parameter (optional):

- P1..Pn weitere Parameter der PUT_RELATION

Anmerkung: Die Parameter P1 bis Pn können in beliebiger Reihenfolge angegeben werden. Die Nummer entspricht dabei der Position des Parameters in dem vergleichbaren Aufruf der PUT_RELATION in einem Eventskript. Soll ein Parameter leer bleiben, muss er mit leerem Inhalt übergeben werden.

Rückgabe:

Keine

Notizen

SoftENGINE
Kaufmännische Softwarelösungen
für Handel, Industrie & E-Commerce

Alte Bundesstraße 16 • 76846 Hauenstein
Telefon: +49 (0) 63 92 - 995 0
www.softengine.de • info@softengine.de