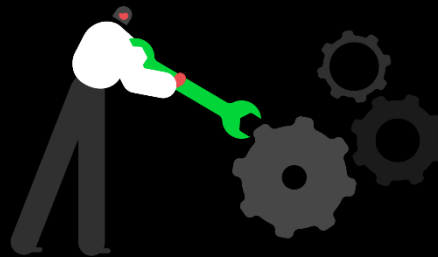




Sage Applikationsserver Administrationshandbuch

# Sage Applikationsserver

Administrationshandbuch



Sage

OHNE AUSDRÜCKLICHE SCHRIFTLICHE ERLAUBNIS DÜRFEN WEDER DAS HANDBUCH NOCH AUSZÜGE DARAUS MIT MECHANISCHEN ODER ELEKTRONISCHEN MITTELN, DURCH FOTOKOPIEREN ODER AUF IRGEND EINE ANDERE ART UND WEISE VERVIELFÄLTIGT ODER ÜBERTRAGEN WERDEN.

IN BEISPIELEN VERWENDETE FIRMEN UND SONSTIGE DATEN SIND FREI ERFUNDEN, EVENTUELLE ÄHNLICHKEITEN WÄREN DAHER REIN ZUFÄLLIG.

DEN IN DIESEM DOKUMENT ENTHALTENEN INFORMATIONEN LIEGT DER ZUR DRUCKLEGUNG AKTUELLE PROGRAMMSTAND ZUGRUNDE. SIE KÖNNEN OHNE VORANKÜNDIGUNG GEÄNDERT WERDEN UND STELLEN KEINE VERPFLICHTUNG SEITENS DES VERKÄUFERS DAR.

URHEBERRECHTLICH GESCHÜTZT – SAGE SOFTWARE GMBH

IN DIESEM BUCH VERWENDETE SOFT- UND HARDWAREBEZEICHNUNGEN SIND ÜBERWIEGEND EINGETRAGENE WARENBEZEICHNUNGEN UND UNTERLIEGEN ALS SOLCHE DEN GESETZLICHEN BESTIMMUNGEN DES URHEBERRECHTSSCHUTZES.

DIE SAGE HANDBUCHREDAKTION IST BEI DER ERSTELLUNG DIESES HANDBUCHES MIT GROßER SORGFALT VORGEHEND. FEHLERFREIHEIT KÖNNEN WIR JEDOCH NICHT GARANTIEREN. SAGE HAFTET NICHT FÜR TECHNISCHE ODER DRUCKTECHNISCHE FEHLER IN DIESEM HANDBUCH. DIE BESCHREIBUNGEN IN DIESEM HANDBUCH STELLEN AUSDRÜCKLICH KEINE ZUGESICHERTE EIGENSCHAFT IM RECHTSSINNE DAR.

SOLLTEN SIE KORREKTUR- ODER VERBESSERUNGSVORSCHLÄGE ZU DIESEM HANDBUCH HABEN, SCHICKEN SIE UNS DIESE BITTE AN DIE E-MAIL-ADRESSE [DOKU@SAGE.DE](mailto:doku@sage.de) ODER VERWENDEN SIE DAS FORMULAR AM ENDE DES HANDBUCHES. WIR BEDANKEN UNS IM VORAUS FÜR IHRE MITARBEIT.

WEITERE INFORMATIONEN ÜBER DIE PRODUKTE VON SAGE SOFTWARE FINDEN SIE IM INTERNET UNTER [HTTP://WWW.SAGE.DE](http://www.sage.de)

FÜR ALLGEMEINE FRAGEN RUND UM UNSERE PRODUKTE UND DIENSTLEISTUNGEN STEHT IHNEN UNSERE KUNDENBETREUUNG UNTER FOLGENDEN KONTAKTMÖGLICHKEITEN GERNE ZUR VERFÜGUNG:

TELEFON: 069-50007-6333

E-MAIL: [KUNDENBETREUUNG@SAGE.DE](mailto:kundenbetreuung@sage.de)

JUNI 2024

# Inhaltsverzeichnis

<b>Konzeptionelle Übersicht .....</b>	<b>4</b>
<b>Warum ist der Applikationsserver erforderlich? .....</b>	<b>4</b>
<b>Verwendungsszenarien .....</b>	<b>5</b>
<b>Fundamentale Konzepte .....</b>	<b>6</b>
Nachrichtenorientierte Kommunikation .....	6
Kommunikationsprotokolle .....	6
Fundamentale Begriffe .....	7
<b>Applikationsserver Architektur .....</b>	<b>8</b>
Kernkomponenten .....	9
Laufzeitkomponenten .....	9
Prinzipielle Funktionsweise .....	9
Konzeptionelle Details .....	10
<b>Installation .....</b>	<b>15</b>
<b>Überblick .....</b>	<b>15</b>
<b>Installationsschritte .....</b>	<b>15</b>
<b>Installierte Komponenten .....</b>	<b>16</b>
<b>Einrichtung von SSL .....</b>	<b>18</b>
<b>SOAP Service-Installation und -Konfiguration .....</b>	<b>19</b>
Assemblies installieren .....	20
Konfigurationseinträge erstellen .....	20
Namespaceregistrierungen vornehmen .....	21
SSL-Zertifikat mit den verwendeten Ports verbinden .....	23
<b>Serviceautorisierung einrichten .....</b>	<b>24</b>
Frontend-Autorisierung .....	24
Autorisierung der Applikation .....	25
<b>SData Endpunkte konfigurieren .....</b>	<b>27</b>
<b>Secure Token Service konfigurieren .....</b>	<b>29</b>
<b>Client-seitiges Loadbalancing für die Sage 100 .....</b>	<b>29</b>
<b>Blobstorage Server konfigurieren .....</b>	<b>30</b>
<b>Administration und Diagnose .....</b>	<b>32</b>
<b>Liveupdate .....</b>	<b>32</b>
<b>Konfiguration .....</b>	<b>32</b>
<b>Tracing .....</b>	<b>46</b>
Standard System.Diagnostics Tracing .....	46
Tracing via Ereignisablaufverfolgung für Windows (ETW) .....	47
Datei-Logging via ETW .....	48
Konfiguration des ETW Datei-Loggings .....	48
Echtzeit-Logging via ETW und TraceLogManager .....	50
<b>Performance-Counter .....</b>	<b>52</b>
<b>Server Manager .....</b>	<b>54</b>

# Konzeptionelle Übersicht

In der heutigen vernetzten und globalisierten Welt ist es erforderlich, Inhouse-Anwendungen über Standards auch nach außen zu öffnen. Um dies zu gewährleisten wird eine Infrastruktur benötigt, die es erlaubt, nach außen öffentliche Services zur Verfügung zu stellen, die intern mit der Anwendung kommunizieren. Die möglicherweise parallel eintreffenden Anfragen müssen auf die jeweilige Anwendungsfunktionalität verteilt, parallel verarbeitet und beantwortet werden. Genau dies stellt der Applikationsserver zur Verfügung.

## » Abschnitte dieses Kapitels

- Warum ist der Applikationsserver erforderlich?
- Verwendungsszenarien
- Fundamentale Konzepte
- Applikationsserver Architektur

## Warum ist der Applikationsserver erforderlich?

ERP-Applikationen wie die Sage 100 sind in der Regel prinzipiell als Rich-Client Applikation ausgelegt. Jegliche Applikationslogik wird im Prozess des Clients ausgeführt und ist für das verwendete Client-DB-Server-Modell optimiert. Nebenläufigkeit muss auf Grund des seriellen, am UI orientierten Prozessmodells, nicht beachtet werden.

Zusätzlich sind solche Applikationen in der Regel statusbehaftet konzipiert. Im Falle der Sage 100 verwendet jeder Client seine eigene Instanz, die über die gesamte Lebenszeit erhalten bleibt. Eine Instanz besteht im Wesentlichen aus einem „Mandant“-Objekt, das wiederum die Verbindung zu den verwendeten Ressourcen hält. Dazu gehören Datenbankverbindungen aber auch Daten, die in einem Cache abgelegt sind. Die Komponenten der Geschäftslogik verwenden dieses Objekt zur Implementierung ihrer Logik.

Eine serverseitige Applikation muss jedoch dazu in der Lage sein, mehrere Anfragen von verschiedenen Benutzern gegen unterschiedliche Ressourcen parallel auszuführen. D.h. hier muss Nebenläufigkeit beachtet werden. Auch wenn die Anfragen in eigenen Instanzen bearbeitet werden, muss jedoch der Zugriff auf gemeinsame Ressourcen des Prozesses synchronisiert werden. Dies betrifft insbesondere statische Variablen und Methoden. Die Implementierung der Synchronisierung von nebenläufigen Threads ist komplex, Fehleranfällig (Deadlocks) und in der bestehenden Anwendung nur mit größtem Aufwand durchführbar.

Die Initialisierung eines Applikationskerns ist relativ teuer im Vergleich zu der Ausführungszeit einer Anfrage. Um zu einem akzeptablen Zeitverhalten zu kommen, ist ein Pooling von statusbehafteten Applikationskernen unerlässlich. Die notwendige Anzahl dieser Kerne mit unterschiedlichem Status sollte jedoch so gering wie möglich gehalten werden, da jede Instanz Ressourcen verbraucht.

Die Hauptaufgaben des Applikationsservers sind also:

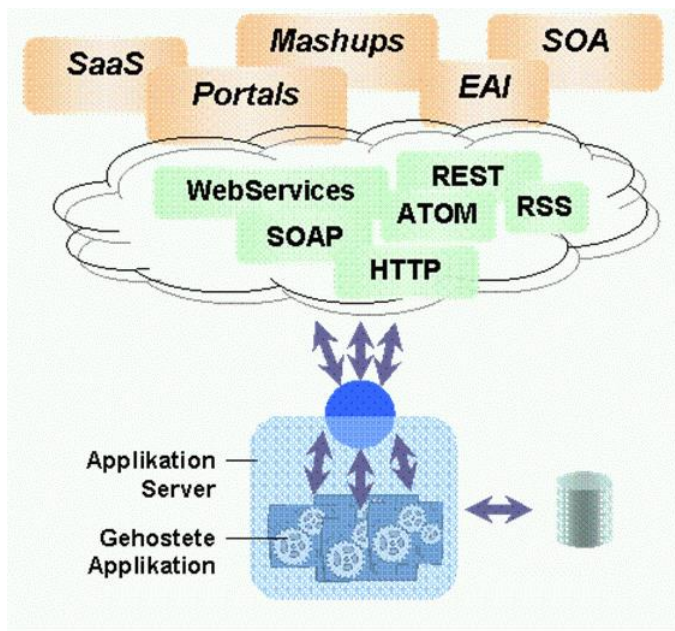
- Isolierung der Anwendung für parallele Ausführung
- Pooling von Applikationskernen

Unter folgenden Rahmenbedingungen:

- Die Umsetzung ist relativ unabhängig von der gehosteten Applikation.
- Die Implementierung von Server-seitiger Logik (Services) sollte genauso einfach möglich sein, wie die heutige, Client-seitige Implementierung von Geschäftslogik. Die Code-Basis ist dabei immer die gleiche.

- Per Applikationsserver kann immer nur genau eine Applikation gehostet werden. (z.B. Sage 100 X.Y)

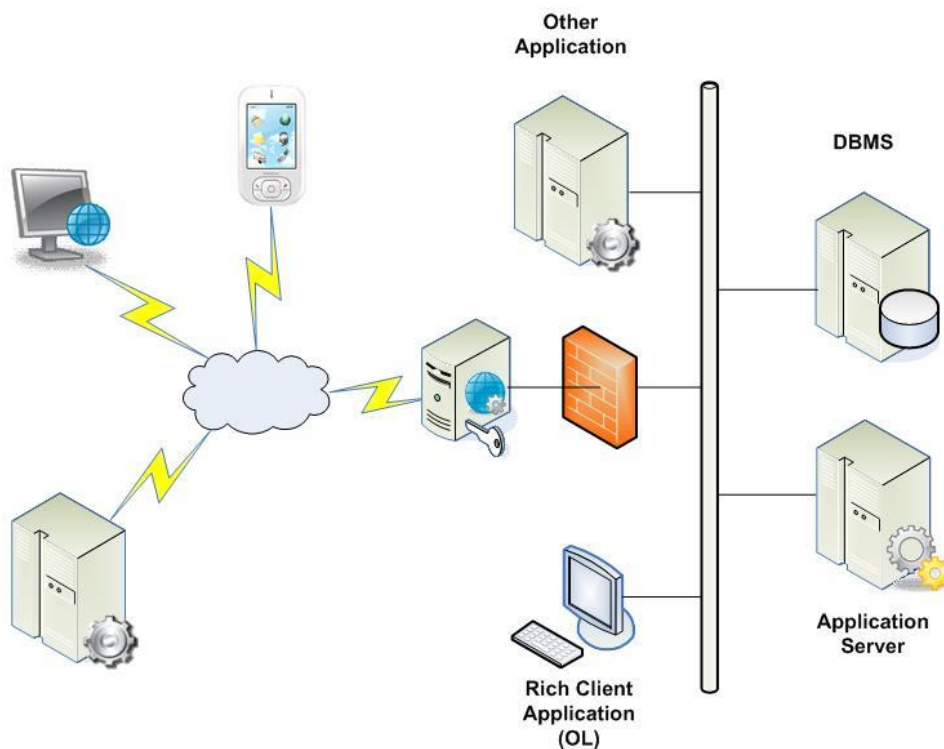
Letztendlich stellt der Applikationsserver Infrastruktur zur Verfügung, auf deren Basis Services entwickelt, gehostet und veröffentlicht werden können.



Das obige Bild zeigt die Hauptaufgabe des Applikationsservers: Die Verarbeitung paralleler Anfragen gegen die gehostete Applikation in isolierten Anwendungsräumen.

## Verwendungsszenarien

Das folgende Bild zeigt ein mögliches Szenario wie der Applikationsserver mit der Sage 100 eingesetzt werden kann.



Innerhalb eines Intranets können Applikationen direkt mit dem Applikationsserver kommunizieren. Dies gilt sowohl

für Clients der gehosteten Applikation aber auch für andere Applikationen.

Anfragen von außen (via Internet) werden von einem Server der DMZ entgegengenommen. Dort wird i.d.R. auch Authentication und Authorization abgebildet. Dort erfolgen dann auch die Umsetzung und das Mapping der veröffentlichten Services gegen die Services, die der Applikationsserver zur Verfügung stellt. Der öffentliche Server in der DMZ leitet den Aufruf über einen geöffneten Port der Firewall an den Applikationsserver weiter. Hier wird die Anfrage bearbeitet und beantwortet. Der in der DMZ gehostete Service setzt die Daten in das geforderte Format um und leitet schlussendlich das Ergebnis zurück zum anfragenden Client.

Beispiele für solche Services sind:

- Lagerbestandsabfrage
- Preisanfrage
- OP-Auskunft
- Beleg-Übermittlung (z.B. Auftrag, Rechnung etc.)
- Kundenselfservices
- Adressänderung
- Änderung der Bankverbindung
- Integrationsservices gegen andere Applikationen

## Fundamentale Konzepte

### Nachrichtenorientierte Kommunikation

Client und Applikationsserver kommunizieren ausschließlich über den Austausch von Nachrichten. D.h. ein Client schickt eine Nachricht mit einer Anfrage an den Server. Der Server bearbeitet diese Anfrage und sendet eine Nachricht als Ergebnis zurück zum Client. Der Applikationsserver verwendet immer das Request-Reply-Muster. Duplex-Verbindungen werden nicht unterstützt.

Nachrichten werden zwischen Endpunkten ausgetauscht. Ein Service kann einen oder mehrere Endpunkte veröffentlichen. Ein solcher Endpunkt ist über sämtliche, zum Austausch einer Nachricht erforderlichen Informationen definiert.

Dies beinhaltet:

- Das Transportprotokoll (http, tcp)
- Das Schema der auszutauschenden Nachricht ()
- Die Formatierung der Nachricht (Text, Binär etc.)

### Kommunikationsprotokolle

Der Applikationsserver unterstützt die Kommunikation sowohl via Soap als auch via SData. SData ist ein von Sage definiertes, öffentliches REST-basiertes Protokoll. Die komplette Spezifikation des SData-Protokolls ist unter <http://sdata.sage.com> zu finden.

# Fundamentale Begriffe

## SOAP-Service

Ein SOAP-Service veröffentlicht einen oder mehrere Endpunkte, die jeweils eine oder mehrere Operationen beinhalten.

## Endpunkt

Zwischen Endpunkten werden Nachrichten ausgetauscht. Ein Endpunkt definiert:

- Die Adresse, unter der ein Service erreicht werden kann
- Der Kommunikationsmechanismus der zum Austausch der Nachrichten verwendet wird (die Bindung)
- Definition der Nachrichten die gesendet und/ oder empfangen werden kann.

## Bindung

Die Bindung definiert, wie mit einem Endpunkt kommuniziert werden kann. Dazu gehören das Transportprotokoll und die Formatierung der auszutauschenden Nachricht. Der Applikation Server stellt eine Sammlung von Bindungen zur Verwendung zur Verfügung.

## Service Vertrag

Ein (SOAP) Service Vertrag fasst mehrere zusammengehörende Operationen eines Services zu einer Einheit zusammen. Ein Service Vertrag ist i.d.R. über ein Interface realisiert.

## Service Operation

Eine Service Operation ist eine Funktion, die eine Funktionalität eines Services implementiert.

## Daten Vertrag

Ein Daten Vertrag beschreibt die Daten, die ausgetauscht werden.

## Hosting

Ein Hosting Prozess bietet den Lebensraum (den Betriebssystem-Prozess → eine .exe), in dem die Services ausgeführt werden können. Er kontrolliert die Lebensdauer der Services. Für den Applikationsserver stehen Windows Service und Konsole als Hosting Prozesse zur Verfügung.

## Client

Ein Client ist ein Konsument von Services, die der Applikationsserver über seine Endpunkte zur Verfügung stellt. Clients können unter unterschiedlichen Technologien implementiert sein.

## Service-Domain

Eine Service-Domain ist das zentrale Element des Applikationsserver. Sie verbindet einen Service und eine Instanz der gehosteten Applikation in einem Prozess-isolierten Container. Parallele Requests werden immer in unterschiedlichen Service-Domains ausgeführt.

## Application Context

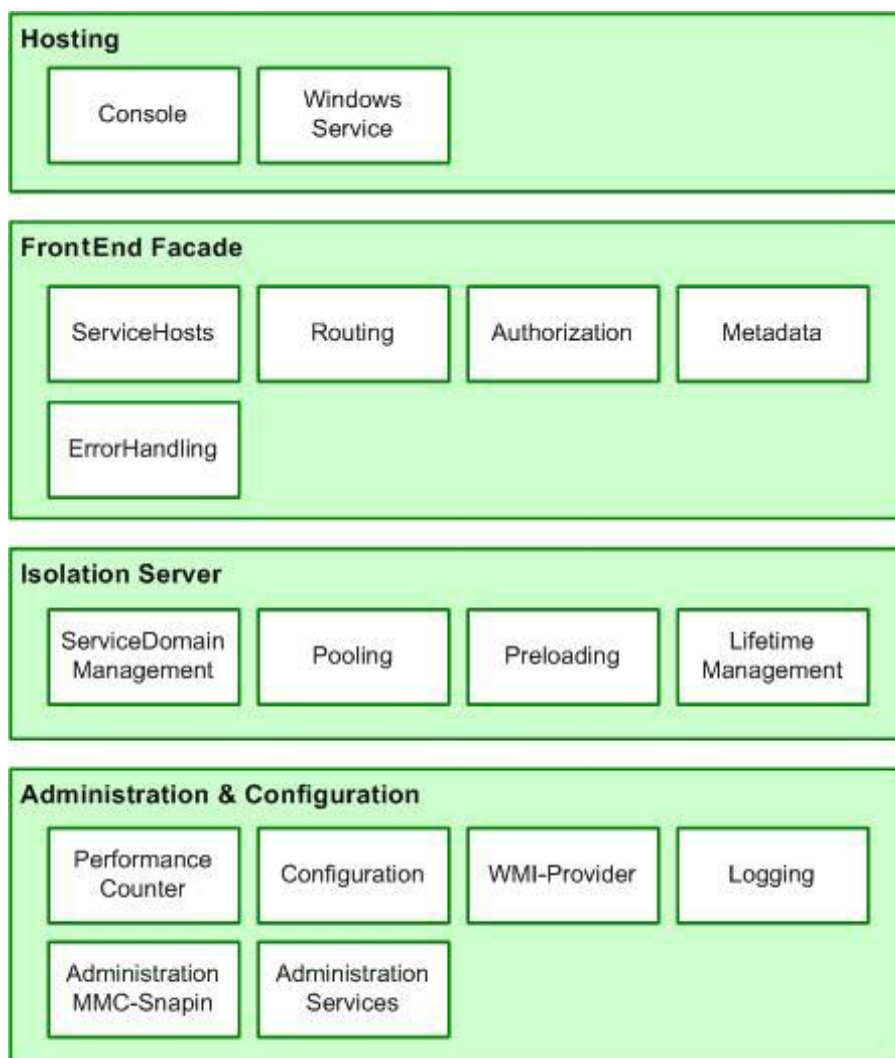
Der Application Context definiert eindeutig den Kontext, unter dem die gehostete Applikation läuft. Applikationsinstanzen mit gleichem Kontext sind austauschbar, d.h. sie verhalten sich für die Serviceimplementierung immer gleich. Der Application Context wird bei jeder Anfrage im Header der Nachricht zum Server übertragen.

## Gehostete Applikation

Der Applikationsserver stellt lediglich Infrastruktur zur Verfügung. Die eigentliche Funktionalität ist in den Service implementiert, die wiederum gegen eine dedizierte Applikation (z.B. Sage 100) arbeiten. Innerhalb des Servers kann nur genau eine solche Applikation verwendet werden.

## Applikationsserver Architektur

Der Applikationsserver besteht aus den folgenden grundlegenden Bestandteilen:





# Kernkomponenten

## Applikationsserver Kern (Frontend Facade und Isolation-Server)

Stellt eine Laufzeitumgebung für die parallele und effiziente Ausführung von Services gegen die gehostete Applikation zur Verfügung.

Für jeden Service (SOAP und/oder SData) wird ein (WCF) Servicehost gestartet, der auf den entsprechenden Ports auf eingehende Nachrichten lauscht.

Stellt für SOAP-Services WSDL-Endpunkte zur Verfügung.

Verwaltet Instanzen der gehosteten Applikation. Stellt jedem Request eine isolierte Applikation für die eigentliche Service-Implementierung zur Verfügung.

Beinhaltet einen Caching-Mechanismus für die verwalteten „Applikationskerne“.

## Host Prozesse

Der eigentliche Applikationsserver Kern benötigt einen Prozess, in dem er läuft. Innerhalb dieses Prozesses wird der Applikationsserver gestartet.

Es stehen 2 Varianten zur Verfügung:

- Windows Service (Sagede.ApplicationServer.WindowsService.exe)  
Für den Betrieb.
- Console (Sagede.ApplicationServer.ConsoleHost.exe)  
Zur Entwicklungszeit, ermöglicht einfacheres Debugging.

Zur jeder Host-Implementierung existiert eine Konfigurationsdatei.

## Administration und Konfiguration

Beinhaltet Werkzeuge zur Verwaltung und Überwachung des eigentlichen Applikationsservers.

# Laufzeitkomponenten

Der Applikationsserver bietet eine Infrastruktur zur Veröffentlichung von Services gegen eine gehostete Applikation. Die weiteren Komponenten zur Laufzeit sind:

## Gehostete Applikation

Beinhaltet die Komponenten der Applikation (Sage 100) und eine spezifische Treiberimplementierung, die vom Applikationsserver Kern für die Ansteuerung der gehosteten Applikation verwendet wird.

## Services

Implementierung der Dienste, die über den Applikationsserver veröffentlicht werden.

Verwenden die gehostete Applikation zur Implementierung der Logik. Dazu wird vom Applikationsserver eine initialisierte Instanz der Applikation zur Verfügung gestellt.

Implementierung der SOAP-Dienste erfolgt auf Basis von WCF.

Implementierung der SData-Dienste erfolgt über das im Applikationsserver enthaltenen SData-Framework.

# Prinzipielle Funktionsweise

Clients und der Applikationsserver interagieren via SOAP und/oder SData Nachrichten. Dabei ist die Technologie des Clients letztendlich unerheblich. Auch nicht Windows Clients können mit dem Server kommunizieren, sofern sie SOAP-Nachrichten via http senden bzw. empfangen können bzw. SData verwenden. Ein Client kann ein anderer Prozess auf derselben Maschine sein, eine andere Windows-Maschine aber prinzipiell auch eine nicht Windows-Maschine, die via Internet mit dem Server kommuniziert.

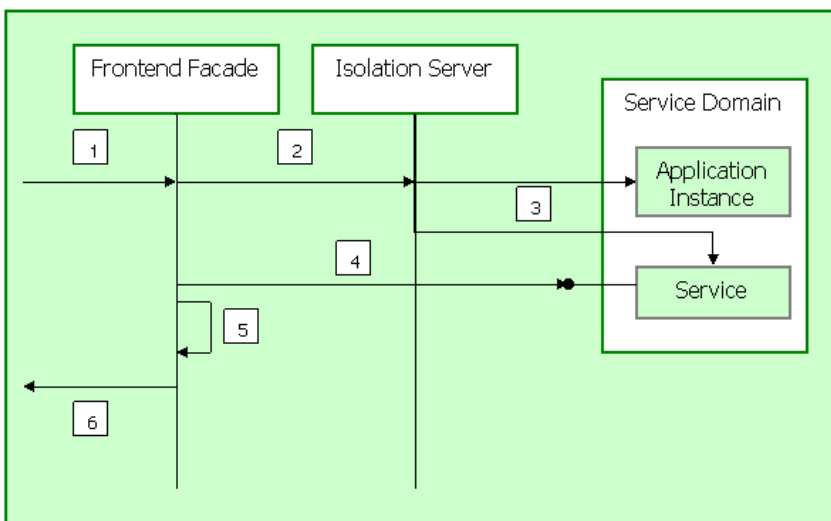
Für die Kommunikation zum Server können bei SOAP-Diensten verschiedene Transportprotokolle verwendet werden. Z.B. http, https (http mit SSL-Transportverschlüsselung) oder aber tcp. Bei der Verwendung von SData, wird die Kommunikation über http bzw. https abgewickelt.

Beim Start des Applikationsserver Host Prozesses werden aus der dazugehörigen Konfigurationsdatei die eingetragenen SOAP Servicedefinitionen ausgelesen und für jeden Service ein Servicehost gestartet. Ein Servicehost registriert Listener auf den entsprechenden Ports, nimmt alle (SOAP) Anfragen entgegen und leitet sie an die Frontend Facade des Applikationsserver Kerns weiter. Daneben wird ein ServiceHost für SData, entsprechend seiner Konfiguration, gestartet.

Die Frontend Facade lässt sich vom Verwaltungsmechanismus des Servers einen Kanal zu einer Service-Domain geben, die einen Service beinhaltet, der die Anfrage beantworten kann. Einer Service-Domain entspricht eine initialisierte Instanz der gehosteten Applikation mit der Implementierung eines Services, die über einen Kanal erreichbar ist. Die gehostete Applikation und Serviceimplementierung stehen exklusiv für eine Anfrage zur Verfügung.

Falls eine passende und freie Service-Domain bereits vorhanden ist, so wird sie wiederverwendet. Im anderen Fall wird eine Neue erzeugt.

Über den erhaltenen Kanal leitet die Frontend Facade die Anfrage an die Service-Implementierung weiter. Das zurückerhaltene Ergebnis wird über den ServiceHost zu dem anfragenden Client zurückgeschickt.



Das Bild zeigt einen kompletten Ablauf eines Requests in der Übersicht:

- Ein Request kommt bei einem Endpunkt der Frontend Facade an.
- Frontend Facade fragt beim Isolation-Server nach einem Service Endpunkt, der den Request beantworten kann.
- Isolation-Server sucht nach einer passenden Service-Domain. Falls nicht vorhanden, so wird eine erzeugt, eine Applikationsinstanz und ein Serviceendpunkt gestartet. Der Service Endpunkt wird an den Aufrufer zurückgegeben.
- Die Frontend Facade bereitet den Request auf und schickt ihn an den von dem Isolation-Server gelieferten Endpunkt. Dort wird der Request bearbeitet und das Ergebnis zurückgeschickt.
- Das Ergebnis wird in das Format des Empfängers konvertiert.
- Das Ergebnis wird zurückgeschickt.

## Konzeptionelle Details

### Anwendungsisolierung und Service-Domains

Die wichtigste Aufgabe der Anwendungsisolierung ist es, für jede Anfrage der Service-Implementierung eine eigene, initialisierte Instanz der gehosteten Applikation transparent zur Verfügung zu stellen. Da die Initialisierung einer Anwendungsinstanz relativ teuer ist, werden vorhandene Instanzen für eine gewisse Lebensdauer gecacht. Nachdem die Anwendungsisolierung eine Anwendungsinstanz erzeugt oder aus dem Cache geholt hat, gibt sie dem Aufrufer einen Endpunkt zurück, an den die Frontend Facade die Anfrage zur Abarbeitung weiterleiten kann.

Das Caching bzw. die Wiederverwendung von Service-Domains ist abhängig von:

- dem Service
- dem Kontext, unter dem die Anwendungsinstanz läuft

D.h. eine Service-Domain kann nur dann wieder verwendet werden, falls diese Angaben der gecachten Service-Domain mit den Angaben der neuen Anfrage übereinstimmen. Der Kontext der Anwendungsinstanz ist abhängig von der gehosteten Applikation.

Im Fall der Sage 100 besteht der Kontext aus:

- User ID
- Passwort
- ID der Datenquelle
- Nummer des Mandanten

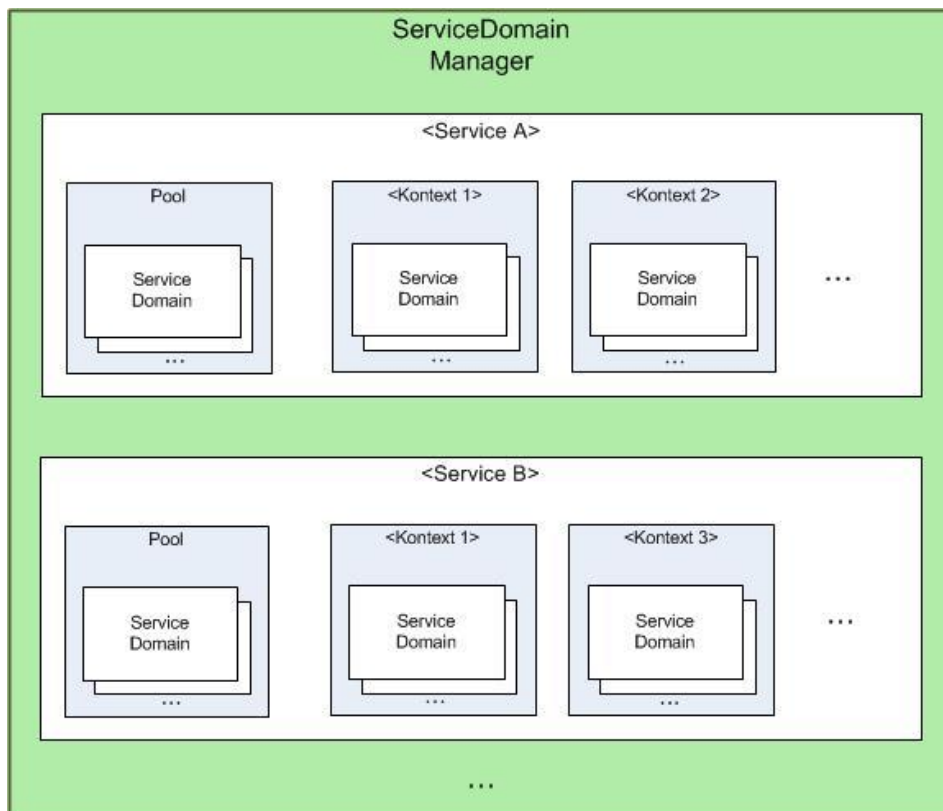
Eine Service-Domain stellt eine isolierte Anwendungsinstanz in Verbindung mit einem Service-Endpoint dar. Eine Service-Domain liegt in einem Isolationsraum, der sicherstellt, dass es keine Verbindung zu anderen Service-Domains gibt. Die eigentliche Isolierung kann über die Verwendung von AppDomains oder aber auch Prozessen erfolgen. Per Default werden Prozesse zur Isolation verwendet. Die Isolation via Prozessen bietet u.a. folgende Vorteile:

- Volle Ausnutzung des vorhandenen Arbeitsspeichers
- Isolation auch von unmanaged Code ist sicher möglich (COM, P/Invoke)

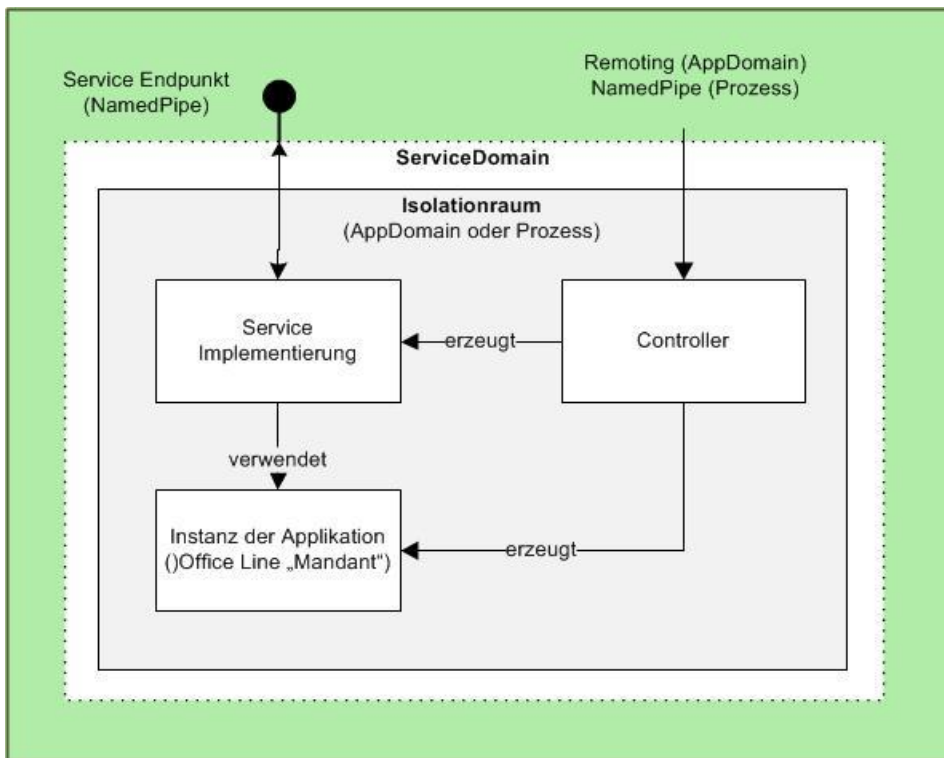
Wird eine neue Service-Domain erzeugt, wird intern eine Singleton-Instanz der gehosteten Applikation erzeugt und initialisiert. Die Serviceimplementierung kann dann im Falle der gehosteten Office Line/Sage 100 über „ServiceContext.Current.Mandant“ auf die Applikation (OL = Mandant) zugreifen.

Innerhalb des Applikationsservers gibt es genau eine Instanz (ServiceDomain Manager), die Service-Domains verwaltet. Da mehrere Request parallel über mehrere Threads abgearbeitet werden, muss die gesamte Implementierung sicher für Multithreading sein.

Der ServiceDomain Manager verwaltet die ServiceDomains in einer Struktur, die folgenden prinzipiellen Aufbau hat:



Service-Domains werden bei Isolation via AppDomains via Remoting bzw. bei Isolation via Prozessen via NamedPipes verwaltet und stellen einen „WCF“-Endpunkt zur Verfügung, über den die eigentliche Serviceimplementierung erreicht werden kann. Der gesamte Isolationsmechanismus ist für Soap und SData identisch.



## Preloading / Pooling

Die Erzeugung einer neuen Service-Domain-Instanz ist vergleichsweise teuer, da in den Isolationsraum die komplette Applikation neu geladen werden muss. Es sind keine Einträge in dem ADO-Connection Pool vorhanden. Zusätzlich muss die komplette Infrastruktur für die Kommunikation mit dem neuen Isolationsraum hochgefahren werden. Aus diesem Grund werden Service-Domains gepoolt.

Beim Start des Servers wird eine konfigurierbare Anzahl von Service-Domains per Service vorerzeugt und in einen Pool gelegt. Muss eine neue Service-Domain zur Laufzeit erzeugt werden, so wird, falls möglich, dazu eine vorinitialisierte Instanz aus dem Pool entnommen. Ist die Lebenszeit einer Service-Domain abgelaufen, so wird sie deinitialisiert und aus der Serviceisolation-Struktur in den Pool verschoben.

Wird eine ServiceDomain einem Pool entnommen und ist eine Mindestanzahl von ServiceDomains des Pools konfiguriert, so wird automatisch im Hintergrund eine neue ServiceDomain erzeugt und damit der Pool wieder aufgefüllt.

## Algorithmus für das Ermitteln einer ServiceDomain

Als erstes wird versucht eine vorhandene, freie ServiceDomain mit dem gleichen Kontext wiederzuverwenden (Service, Benutzer, Datenbank, Mandant).

Schlägt dies fehl, wird als nächstes versucht eine ServiceDomain mit änderbarem Kontext zu finden. Im Falle der Sage 100 ist ein Kontext änderbar, falls lediglich der Benutzer unterschiedlich ist. Eine solche ServiceDomain wird jeweils dem Kontext mit den meisten ServiceDomains entnommen. Dabei verbleibt aber immer eine ServiceDomain bei dem Kontext. Wird eine solche ServiceDomain gefunden, wird ein Benutzerwechsel durchgeführt.

Konnte keine wiederverwendbare ServiceDomain ermittelt werden, wird jetzt versucht eine neue zu erzeugen. Sofern im Pool noch eine ServiceDomain vorhanden ist, wird zunächst diese verwendet.

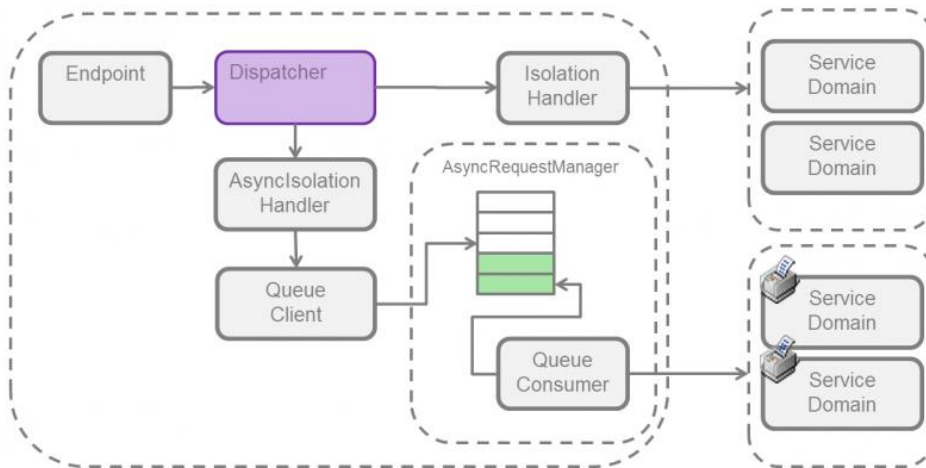
War auch dies nicht erfolgreich wird als nächstes versucht eine nicht kompatible, freie ServiceDomain für den Service zu finden. Wird eine solche gefunden, so wird sie entsprechend dem neuen Kontext neu initialisiert.

Schlägt auch dies fehl, wird zum Schluss versucht eine freie ServiceDomain eines anderen Services zu verwenden. Dabei wird zunächst der Pool, dann erst die aktive Struktur durchsucht. Falls auch jetzt keine ServiceDomain ermittelt werden konnte, erhält der Client eine Fehlermeldung, daß der Server ausgelastet ist.

## Asynchrone SData Anfragen

Asynchrone SData Anfragen (Z.B. Reports in der Sage 100 oder Sage CRM Integration) werden Server-seitig asynchron abgearbeitet. Dazu wird ein eigener Mechanismus mit dedizierten ServiceDomains verwendet. Die Anfragen werden zunächst in eine Warteschlange gestellt. Eine Komponente schaut periodisch nach, ob Anfragen in der Warteschlange

stehen. Ist dies der Fall, werden sie zur Abarbeitung an ServiceDomains übergeben, die ausschließlich für diesen Zweck vorgehalten werden.



## Security

### Transfer Security

Transfer Security soll sicherstellen, dass Nachrichten vom Client zum Server geschickt werden können, ohne dass:

- der Inhalt geändert werden kann
- ein unberechtigter Dritter sie lesen kann

Außerdem muss sichergestellt werden, dass Client bzw. Server der ist, der er vorgibt zu sein (Authentisierung).

Innerhalb des Applikationsservers kann Transfer Security über die Verwendung von SSL sichergestellt werden. Dazu müssen die entsprechenden Bindungen verwendet werden. Achtung: Auf produktiv-Systemen sollte immer SSL verwendet werden!

### Authentisierung auf Anwendungsebene

Zusätzlich muss sich ein Client bei der gehosteten Applikation anmelden. Die Authentisierung auf Anwendungsebene ist i.d.R. im Applikationsspezifischen Treiber des Applikationsservers implementiert. Für den Treiber der Sage 100 geschieht dies bei der Initialisierung der Applikation.

### Autorisierung

Bei der Autorisierung wird sichergestellt, dass der anfragende Client (bzw. der anfragende Benutzer) diese Operation dieses Services ausführen darf. Autorisierung kann auf der Ebene des Applikationsservers (Frontend Autorisierung) unabhängig von der gehosteten Applikation und/oder auf der Ebene der gehosteten Applikation (Application Autorisierung) erfolgen.

- Frontend Autorisierung  
Bei der Frontend Autorisierung wird validiert, ob der anfragende Benutzer die Aktion (Operation) auf diesem Service ausführen darf. Die Überprüfung erfolgt direkt bei Eingang in den Applikationsserver. In der mitgelieferten Beispielimplementierung werden die Zugangsberechtigungen in einer XML-Datei (Sagede.ApplicationServer.Authorization.xml) verwaltet. Es kann ein eigener Provider entwickelt werden, der diese Daten aus einer anderen Datenquelle liest. Nicht autorisiert werden können: Datenquelle und Mandant. Frontend Autorisierung steht nur für Soap-basierte Dienste zur Verfügung.
- Application Autorisierung  
Bei der Application Autorisierung ist es die Aufgabe der gehosteten Applikation die Zugangsberechtigung zu validieren. Die Überprüfung erfolgt direkt bevor die Anfrage an die Serviceimplementierung weitergeleitet wird. Im Fall der Sage 100 wird gegen das Autorisierungssystem der Office Line bzw. Sage 100 geprüft. Die entsprechenden (Update) Berechtigungen müssen mit dem Sage-100-Administrator vergeben werden.
- SData  
Die Autorisierung von SData-Anfragen erfolgt über den Dienst selbst.

## Schutz vor Attacken

Nach 3-maliger erfolgloser Anmeldung werden der anfragende Benutzer und / oder die IP-Adresse des Aufrufers für eine konfigurierbare Zeit gesperrt (per default: 10 Minuten).

## Secure Token Service

Für SData Endpunkte verfügt der Applikationsserver über einen internen Secure Token Service (STS). Aufgabe des STS ist die Authentisierung eingehender Anfragen. Als Ergebnis einer erfolgreichen Authentisierung liefert der STS ein signiertes Token. Anschließend wird die Anfrage via ServiceDomain an die gehostete Applikation weiter geleitet.

## Administration/Monitoring

Im Vergleich zu üblichen Desktop-Programmen bietet ein Server kein User-Interface. Damit ist ein Einblick in die Abläufe und eine Beeinflussung derselben nicht ohne weiteres möglich. Während ein normales Programm von der Interaktion mit dem Benutzer lebt, und dieser über das UI Rückmeldung über den Erfolg oder Misserfolg bekommt und den Status des Programms jederzeit vor Augen hat, läuft ein Server unsichtbar im Hintergrund und tritt immer erst auf Anforderung verschiedenste Clients in Aktion.

Aus diesem Grund gibt es spezielle Wege um einen Server zu beobachten und zu verwalten:

- Administrationsclient (MMC-Snapin)
- Logging
- Scripting

### Logging

Mittels Logging protokolliert der Server die wichtigsten Aktivitäten und eventuell auftretende Fehler. Es liefert einen Einblick in die Auslastung des Servers, in seine Stabilität oder Fehleranfälligkeit. Im Fehlerfall gibt es die Möglichkeit zusätzliche Informationen für das Logging freizuschalten und so den Weg, der zu einem Fehler oder einem Problem führt nachvollziehen zu können.

Logging wird im Applikationsserver auf zwei Arten ausgeführt, zum einen über die Protokollierung von Ereignissen. Dies passiert direkt aus dem Server heraus. Wichtig für eine automatisierte Auswertung der Protokollierung ist die Einhaltung eines einheitlichen Textformats für die Logeinträge. Zur Auswertung solcher Logs gibt es frei erhältliche Parser. Ein Beispiel ist der Microsoft Log Parser. Mit ihm kann man in einer SQL ähnlichen Syntax Logdateien abfragen. Die Protokollierung erfolgt per Default über die Event Tracing für Windows Technologie. Für weitere Details siehe das Kapitel „Tracing“ in „Administration und Diagnose“.

Ein zweiter Mechanismus für das Logging sind Performance-Counter. Diese werden ebenfalls innerhalb des Applikationsservers zu bestimmten Zeitpunkten aufgerufen um Informationen zu protokollieren oder in den Speicher aufzunehmen. Diese Performance-Counter können über den Performance Monitor von Microsoft (perfmon.exe), der ein MMC-Plugin ist, abgerufen werden. Der Performance Monitor liefert, wenn er aufgerufen wird Livedaten aus dem „Inneren“ des Servers.

# Installation

## » Abschnitte dieses Kapitels

- Überblick
- Installationsschritte
- Installierte Komponenten
- Einrichtung von SSL
- SOAP Service-Installation und -Konfiguration
- Serviceautorisierung einrichten
- Client-seitiges Loadbalancing für die Sage 100

## Überblick

Ein lauffähiges Gesamtsystem setzt sich zusammen aus:

- Dem Applikationsserver, der die Infrastruktur bereitstellt.
- Services, die über den Applikationsserver veröffentlicht werden und intern gegen die gehostete Applikation arbeiten.
- Gehosteten Applikationen, für die Services über den Applikationsserver veröffentlicht werden.

Die Installation des Applikationsservers erfolgt über das Setup der gehosteten Applikation.

## Installationsschritte

- Applikationsserver über das Setup der Applikation installieren

Mit der Installation werden folgende Schritte automatisch durchgeführt:

- Installation von Zertifikaten für SData SSL-Endpunkte
- Namespace Reservierungen für die SData Endpunkte
- Konfiguration der SData-Endpunkte in „Sagede.ApplicationServer.SData.config“
- Verbindung der SSL SData Ports mit dem Zertifikat
- Optional für SOAP Services:
  - SOAP Services, soweit vorhanden, installieren
  - ACL der http-Ports konfigurieren
  - SSL konfigurieren

## Achtung

Für den Namen des Rechners, auf dem der Applikationsserver installiert ist, dürfen ausschließlich zugelassene Zeichen (0-9, a-z und "-") verwendet werden, keine Sonderzeichen, da sonst der Applikationsserver nicht läuft. Bei Installation des Betriebssystems wird darauf hingewiesen, im laufenden Betrieb kann der Rechnername jedoch auch mit Sonderzeichen im Namen geändert werden. Zwar wird auch dann eine Warnung ausgegeben, diese kann man jedoch ignorieren (mit der Folge, dass nach der Namensänderung der Application Server nicht mehr läuft).

## Hinweis

Die mitgelieferten bzw. erzeugten Zertifikate sind ausschließlich für die Verwendung im Intranet. Soll direkt aus dem Internet auf den Applikationsserver zugegriffen werden können, ist ein gültiges, von einer Zertifizierungsstelle ausgestelltes Zertifikat zu verwenden.

Die Lizenz des Applikationsservers liefert der Treiber der gehosteten Applikation. D.h. im Falle der Sage 100 muss die Lizenz der Sage 100 eine Lizenz des Applikationsservers beinhalten. Der Name des zu verwendenden Mehrbenutzerdienstes wird ebenfalls von der Applikation geliefert. Um SOAP-Services zu veröffentlichen, die nicht von Sage signiert wurden, ist eine „Enterprise Server“ Lizenz erforderlich.

# Installierte Komponenten

## Startmenü

Aus dem Startmenü des Applikationsservers lassen sich folgende Funktionen aufrufen:

- Anzeige des Handbuchs
- Start des Dialogs für die Basiskonfiguration
- Start des Liveupdates
- Anzeige des MMC-Snapins
- Start der Powershell mit geladenem Modul des Applikationsservers

## Applikationsserver Kern

Komponente	Pfad	Beschreibung
Sagede.ApplicationServer.dll	Root	Kern des Applikationsservers.
Sagede.ApplicationServer.Communication.dll	Root	Von Applikationsserver und Client-assemblies gemeinsam verwendete Klassen.
Sagede.ApplicationServer.ConsoleHost.exe	Root	Hostanwendung für den Applikationsserver in einer Konsole.
Sagede.ApplicationServer.ConsoleHost.exe.config	Root	Konfigurationsdatei für den ConsoleHost
Sagede.ApplicationServer.WindowsService.exe	Root	Hostanwendung für den Applikationsserver als Windows Dienst
Sagede.ApplicationServer.WindowsService.exe.config	Root	Konfigurationsdatei für den WindowsService Host.
Sagede.ApplicationServer.IsolationProcess.exe	Root	Host, für die Isolation von ServiceDomains via Prozessen
Sagede.ApplicationServer.TrackingStorageService.dll	Root	Stellt einen Speicher für asynchrone SData Ergebnisse zur Verfügung



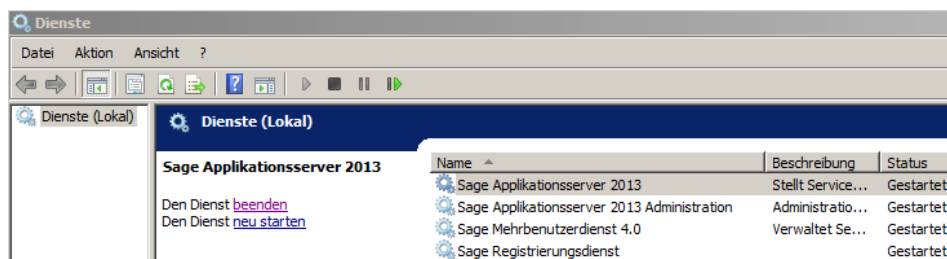
MultiUserServiceClient.dll	Root	Client für den Sage Mehrbenutzerdienst
Sagede.ApplicationServer.Authorization.xml	Root	SOAP Frontend Autorisierungsdaten
Sagede.ApplicationServer.chm	Root	Handbuch / Hilfe
Sagede.Shared.ServiceModel.dll	GAC	Grundlegende Klassen für Server und Services
Sagede.Shared.Core.dll	GAC	Systemklassen
<b>Fremdkomponenten</b>		
System.Data.SQLite.dll	Root	SQLite embedded Datenbank zur Speicherung von asynchronen SData Ergebnissen
Antlr3.Runtime.dll	GAC	Hilfsbibliothek für das Parsen von SData Url's
Newtonsoft.Json.dll	GAC	Hilfsbibliothek für die Serialisierung von SData Nachrichten via JSON

## Applikationsserver Administration

Komponente	Pfad	Beschreibung
Sagede.ApplicationServer.Administration.dll	Root	Basisklassen für die Administration
Sagede.ApplicationServer.Administration.Client.dll	Root	Basisklassen für Administrationsclients
Sagede.ApplicationServer.Administration.Service.exe	Root	Hostanwendung für die Administration des Applikationsservers in einem Windows Dienst.
ASADMIN.exe	Root	Enthält den Basis-Konfigurationsdialog und die Konsole, über die Kommandos ausgeführt werden können.

## Dienste

Nach der Installation sind auf dem Server folgende Dienste vorhanden und müssen zur Laufzeit gestartet sein.



## Achtung

**Läuft der Applikationsserver Windowsdienst unter einem minderpriviligierten Account und soll tcp mit Transportsicherheit verwendet werden (für SOAP-Services), so muss sichergestellt werden, dass dieser Account Zugriffsrechte auf den Schlüssel des verwendeten Zertifikats hat. Zur Erstellung eines Zertifikats siehe: „Einrichtung von SSL“.**

## Einrichtung von SSL

Zur Einrichtung einer verschlüsselten Verbindung zwischen Client und Server via SSL ist ein SSL-Zertifikat auf Serverseite erforderlich. Ein Zertifikat wird von einer Certification Authority (z.B. Versign) ausgestellt. Für interne Testzwecke kann jedoch ein eigenes Zertifikat erzeugt werden.

Das Zertifikat ist im Zertifikatsspeicher der lokalen Maschine im Ordner „Eigene Zertifikate“ (Englisch: „Personal“) zu speichern. Anschließend muss der abzusichernde Port mit dem Zertifikat verbunden werden.

Für SData-Services via https werden bereits beim Setp des Servers alle notwendigen Schritte durchgeführt.

## How To: MMC-Snapin für die Verwaltung von Zertifikaten öffnen

### • MMC starten:

- Start
- Ausführen
- „MMC“
- Enter

### • Zertifikat-Snapin öffnen:

- Datei

- Snapin hinzufügen / entfernen
- Hinzufügen
- Doppelklick auf Zertifikate
- „Computerkonto“ auswählen
- Weiter
- Fertig stellen
- Schließen
- OK

## Erstellung eines „eigenen“ Zertifikats

### Infos

Details zur Erstellung eines eigenen Zertifikats finden Sie bei Microsoft:

- zu „makecert“-Aufrufen via VS-Command-Prompt unter <http://msdn2.microsoft.com/en-us/library/bfskty3.aspx>
- zu „httpcfg“-Aufrufen via VS-Command-Prompt unter <http://msdn2.microsoft.com/en-us/library/ms733791.aspx>

### Vorgehensweise:

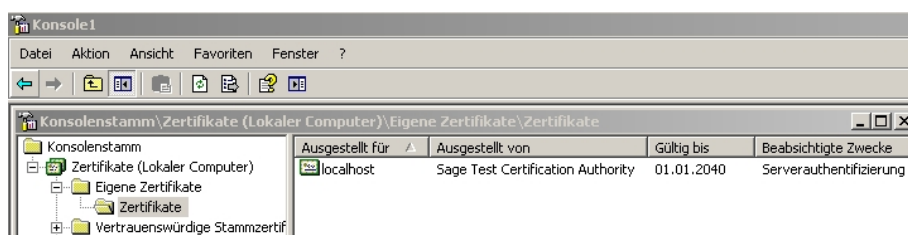
- Zertifikat einer CA erstellen:
  - Skript „CA-Zertifikat erstellen“:
 

```
makecert -sv SignRoot.pvk -cy authority -r signroot.cer -a sha1 -n "CN=Sage Test Certification Authority" -ss my -sr localmachine
```
- Zertifikat via MMC-Snapin von „Eigene Zertifikate“ („Personal“) in „Vertrauenswürdige Stammzertifizierungsstellen“ kopieren („Trustet Root Certification Authorities“).
- SSL-Zertifikat erstellen:
  - Skript „SSL-Zertifikat erstellen“:
 

```
makecert -iv SignRoot.pvk -ic signroot.cer -cy end -pe -n CN="localhost" -eku 1.3.6.1.5.5.7.3.1 -ss my -sr localmachine -sky exchange -sp "Microsoft RSA SChannel Cryptographic Provider" -sy 12
```

Erzeugt ein Zertifikat für <https://localhost>. Für einen anderen Rechnernamen ist „localhost“ durch den DNS-Name zu ersetzen.

Das mit „makecert“ erstellte Zertifikat befindet sich jetzt bereits im Zertifikatsspeicher:



## SOAP Service-Installation und -Konfiguration

Um SOAP Services auf dem Applikationsserver zu veröffentlichen sind folgende Schritte erforderlich:

- SOAP Service Assemblies installieren.
- Konfigurationseinträge der installierten SOAP-Services in den Konfigurationsdateien des Applikationsservers für jeden verwendeten Host einfügen.
- Namespace Reservierungen vornehmen, sofern der Host nicht unter einem Admin-Account läuft.
- SSL-Zertifikat mit den verwendeten Ports verbinden, falls SSL-Ports verwendet werden, die noch nicht verbunden sind.
- SOAP Service Autorisierung einrichten, soweit erforderlich.
- Server neu starten.

## Assemblies installieren

Service Assemblies sind die Assemblies, die den Servicevertrag definieren und die Assemblies, die den Servicevertrag implementieren.

Zur Installation sind alle Assemblies in das Binär-Verzeichnis der gehosteten Applikation zu kopieren.

## Konfigurationseinträge erstellen

Die zu veröffentlichenden Services sind dem Applikationsserver über Einträge in die Konfigurationsdatei bekannt zu machen:

- Die Daten können manuell editiert oder mit Hilfe des Programms „ASADMIN.exe“ generiert werden.
- ASADMIN.exe muss in einem Command-Prompt ausgeführt werden.
- ASADMIN.exe befindet sich im Installationsverzeichnis des Servers.

### **Skript: Verwendung von ASADMIN.exe zur Generierung von <service>-Tags**

#### **Aufrufsyntax:**

**ASADMIN.exe /command:scripts /r:<Basisverzeichnis> [/a:OL /? [/q]] <Assembly 1-n>**

#### **/r:<Basisverzeichnis>**

Das Verzeichnis, in das die Skripts erzeugt werden.

#### **/a:<Gehostete Applikation>**

Sage 100: Zusätzlich werden SQL-Skripte für die Office-Line-bzw. Sage-100-Autorisierung erzeugt.

#### **<Assembly 1-n>**

Der volle Dateiname inklusive Pfad der Assemblies, die untersucht werden sollen. Muss am Ende stehen. Trennung mehrere Assemblies via Leerzeichen.

#### **/?**

Anzeige der Hilfe.

#### **/q**

Keine Ausgabe von Meldungen.

#### **Beispiel**

**asadmin.exe command:/scripts /a:OL /r:"c:\test" "c:\bin\contracts1.dll" "c:\bin\services1.dll"**

→ Untersucht die angegebenen Assemblies

→ Erzeugt die Skripts in das Verzeichnis „c:\test“

Erzeugte Servicekonfiguration:

```

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<service name="ArtikelService" faultPromotion="None"
  applicationAuthorization="true
  namespace="http://Sagede/ApplicationServer/OfficeLine/Samples">

  <contract
    type="Sagede.ApplicationServer.OfficeLine.Samples.IArtikelService,
    ArtikelContracts, Version=1.0.0.0, Culture=neutral,
    PublicKeyToken=null" />
  <implementation
    type="Sagede.ApplicationServer.OfficeLine.Samples.ArtikelService,
    ArtikelService, Version=1.0.0.0, Culture=neutral,
    PublicKeyToken=null" />
</service>

```

Das gesamte <service>-Tag ist in die Konfigurationsdatei „Sagede.ApplicationServer.Soap.config“ in das Tag <services> zu kopieren.

Weitere Informationen zu den Einstellungsmöglichkeiten in der Konfigurationsdatei sind im Hilfethema „Administration und Diagnose“ zu finden.

Zur direkten Installation kann ASADMIN.exe mit dem Parameter „/i“ verwendet werden. Dann werden die angegebenen Assemblies in das Services Verzeichnis des Applikationsservers kopiert und die Registrierungseinträge direkt in die Konfigurationsdatei des Servers geschrieben.

## **Skript: Verwendung von ASADMIN.exe zur direkten Installation von SOAP Services**

**Aufrufsyntax: ASADMIN.exe /command:i [/a:(0|1)] [/?] [/q] <Assembly 1-n>**

### **/a:(0|1)**

- 0: Die Autorisierung auf Applikationsebene wird in der Konfiguration für die zu registrierenden Services ausgeschaltet.

Falls "1" oder nicht angegeben, wird die Autorisierung auf Applikationsebene eingeschaltet.

### **<Assembly 1-n>**

Der volle Dateiname inklusive Pfad der Assemblies, die untersucht werden sollen. Muss am Ende stehen. Trennung mehrerer Assemblies via Leerzeichen.

### **/?**

Anzeige der Hilfe.

### **/q**

Keine Ausgabe von Meldungen.

## **Namespaceregistrierungen vornehmen**

HTTP.SYS lässt nicht zu, dass Nicht-Administratoren Listener ohne ausdrückliche Genehmigung eines Administrators registrieren. D.h., läuft der Server unter einem minderprivilegierten Account (was in der Regel der Fall sein sollte), so muss die ACL von http.sys angepasst werden. Dies ist prinzipiell für jede URI eines Services erforderlich. Ein Administrator muss der Anwendung in HTTP.SYS eine Namespacereservierung gewähren. Dadurch sagt der Administrator im Wesentlichen "Dieser Benutzer darf über HTTP auf ein URL-Präfix warten, das ich festlege". Ein Administrator kann diese Abhörberechtigungen einem einzelnen Benutzer oder einer Gruppe gewähren.

## How to: Verwendung von httpcfg.exe (Vor Vista) zur Namespacereservierung

### Namespacereservierungen auflisten:

**httpcfg query urlacl**

### Namespace reservieren:

**httpcfg set urlacl -u http://+:8080/MyServices/ -a D:(A;;GX;;;S-1-5-21-1681502023-2202157333-1552196959-1028)**

- Der Namespace URI muss mit einem "/" abgeschlossen werden. Dabei spielt keine Rolle, ob es sich um einen absoluten oder relativen URI handelt.
- Der URI hat immer das Format: **http[s]://+:[PORT]/[Service]/**
- Der Parameter der „-a“ folgt, steht für die Zugriffssteuerungsliste (ACL), die in SSDL (Security Description Definition Language) angegeben ist. Dabei stellt der letzte Parameter der semikolon-separierten Liste die SID des Accounts dar, unter dem der Applikationsserver Dienst läuft.

## How to: Verwendung von netsh.exe (Vista+) zur Namespacereservierung

### Namespace reservieren:

**netsh http add urlacl url=https://+:port/Services user=authority\serviceaccount**

Die Namespacereservierung ist für jede verwendete URL zu wiederholen.

## How to: Verwendung von ASADMIN.exe zur Verwaltung von Namespacereservierungen

### Namespacereservierungen auflisten:

**ASADMIN.exe /command:urlacl /action:show**

### Namespacereservierung entfernen:

**ASADMIN.exe /command:urlacl /action:delete /url:[Url-Prefix]**

**Beispiel: ASADMIN.exe /command:urlacl /action:delete /url:https://+:7771/myervice**

### Namespacereservierung hinzufügen:

**ASADMIN.exe /command:urlacl /action:add /url:[Url-Prefix]/accounts:[Accounts im SDDL Format]**

**Beispiel: ASADMIN.exe /command:urlacl /action:add /url:https://+:7771/myervice /accounts:SY,AU**

Hinweise:

Berechtigt die angegebenen Benutzer auf der Url abzuhören.  
accounts:[Komma-seperierte List mit Benutzer im SDDL-Format]  
Falls nicht angegeben wird SY verwendet.

Benutzer im SDDL Format:

SY – NT-Autorität\SYSTEM

NS – NT-Autorität\NETZWERKDIENT

AU – NT-AUTORITÄT\Authentifizierte Benutzer

WD – Jeder

LS – VORDEFINIERT\Benutzer

BA – VORDEFINIERT\Administratoren

BU – VORDEFINIERT\Hauptbenutzer

# SSL-Zertifikat mit den verwendeten Ports verbinden

Werden Bindungen mit Transportsicherheit via SSL verwendet, so muss jeder Port zur Verwendung mit SSL konfiguriert werden. Dazu ist ein entsprechendes Zertifikat im Zertifikatsspeicher der lokalen Maschine erforderlich (siehe „Einrichtung von SSL“).

## Beispiel

### Endpunkte:

`https://mydnsname:6665/myservices1`

`https://mydnsname:6666/myservices2`

Es müssen die Ports 6665 und 6666 mit einem Zertifikat verbunden werden.

Um ein Zertifikat mit einem Port zu verbinden, sind folgende Angaben erforderlich:

- Der Port, auf dem der Service einen Listener hat → [PORT]
- Der Thumbprint des Zertifikats (ohne Leerzeichen). → [THUMBPRINT]

## How to: Ermittlung des Thumbprints

- Via MMC aus dem Zertifikat kopieren:
  - Doppelklick auf <Zertifikat/Details/Fingerabdruck (Thumbnail)>
  - Ergebnis z.B.: „7d 76 82 e4 b0 43 24 95 da 57 30 45 85 04 43 c9 11 c4 20 a7“
- Enthaltene Leerzeichen entfernen:
  - Ergebnis z.B.: „7d7682e4b0432495da573045850443c911c420a7“
- Eine GUID im „Registry Format“ → [GUID], z.B.: „{A89BF85C-CEAF-465e-B091-ADEECFF29C30}“

Zur Verbindung eines Ports mit einem Zertifikat werden dienen Kommandozeilen-Programme, die in einem Command-Prompt aufgerufen werden müssen:

### Skript: Port mit Zertifikat verbinden (Vor Vista)

```
httpcfg set ssl /i 0.0.0.0:[PORT] /h [THUMBPRINT] /g "[GUID]"
```

### Skript: Port mit Zertifikat verbinden (Vista+)

```
netsh http add sslcert ipport:0.0.0.0:[PORT] certhash=[THUMBPRINT] appId=[GUID]
```

### Beispiele „Port mit Zertifikat verbinden“:

[PORT] = 6665

[THUMBPRINT] = 7d7682e4b0432495da573045850443c911c420a7

[GUID] = {A89BF85C-CEAF-465e-B091-ADEECFF29C30}

Vor Vista:

```
httpcfg set ssl /i 0.0.0.0:6665 /h 7d7682e4b0432495da573045850443c911c420a7 /g "{A89BF85C-CEAF-465e-B091-ADEECFF29C30}"
```

Ab Vista:

```
netsh http add sslcert ipport: 0.0.0.0:6665 certhash=7d7682e4b0432495da573045850443c911c420a7 appId={A89BF85C-CEAF-465e-B091-ADEECFF29C30}
```

Diese Aktion ist für jeden verwendeten Port zu wiederholen.

## How to: Konfigurierte Ports auflisten / löschen via httpcfg

### Konfigurierte Ports anzeigen:

```
httpcfg query ssl
```

### Konfigurierte Ports löschen:

```
httpcfg delete ssl -i 0.0.0.0:7779
```

### Zusätzliche Infos:

- Zur Verwendung von netsh siehe: <http://support.microsoft.com/kb/242468>.
- Die „httpcfg.exe“ kann als Bestandteil von [Windows XP SP2 Support Tools](#) heruntergeladen werden.

## How to: Konfigurierte SSL Ports via ASADMIN.exe auflisten / zuordnen / löschen

### Konfigurierte Ports anzeigen:

```
ASADMIN.exe /command:sslbinding /action:show
```

### Konfigurierte Ports löschen:

```
ASADMIN.exe /command:sslbinding /action:delete /port:[port]
```

### SSL-Bindung erstellen:

```
ASADMIN.exe /command:sslbinding /action:add /port:[port] /certhash:[Thumbprint des Zertifikats]
```

### Deinstallation von Services

Zur Deinstallation von SOAP-Services müssen die entsprechenden Einträge aus den Konfigurationsdateien entfernt werden. Nach einem Neustart des Servers sind die Services dann nicht mehr erreichbar.

Wurde für die zu deinstallierenden Services Namespacereservierungen vorgenommen, können diese via httpcfg / netsh bzw. ASADMIN.exe wieder entfernt werden. Das gleiche gilt für die SSL Ports.

## Serviceautorisierung einrichten

### Frontend-Autorisierung

Das Ein- bzw. Ausschalten der Frontend Autorisierung erfolgt in der Konfigurationsdatei auf Server-Ebene. Soll Frontend Autorisierung verwendet werden, muss ein `authorizationManager` Tag in „Sagede.ApplicationServer.Core.config“ vorhanden sein und das Attribut `enabled` auf `"true"` stehen.

### Ein- /Ausschalten der Frontend Autorisierung in der Konfiguration

```
...  
<security>  
  <authorizationManager  
    provider="Sagede.ApplicationServer.Security.  
      XmlFileServiceActionAuthorizationDataProvider,  
      Sagede.ApplicationServer"  
    connection="Sagede.ApplicationServer.Authorization.xml"  
    enabled="true"  
  />  
</security>  
...
```

### Erläuterungen:

#### Provider

Wird der Default-Provider verwendet, so sind entsprechend den zu berechtigenden Services bzw. User Einträge in den Ressourcen vornehmen, die die Autorisierungsdaten beinhalten.



## connection

Die xml-Datei, die vergebene Rechte beinhaltet, ergibt sich aus dem Attribut **connection**. Per Default ist der Name der Datei "[Sagede.ApplicationServer.Authorization.xml](#)" und befindet sich im Root-Verzeichnis des Applikationsservers.

## Beispiele: Sagede.ApplicationServer.Autorization.xml

### Alle User haben Zugriff auf alle Operationen aller Services:

```
<?xml version="1.0" encoding="utf-8" ?>
<sagede.applicationServer.authorization.policy>
  <allow name="*" service="*" action="*" />
</sagede.applicationServer.authorization.policy>
```

### User Sage hat alle Rechte; User Test hat nur das Recht auf die Operation „GetInfo“ des Services „BelegService“>

```
<?xml version="1.0" encoding="utf-8" ?>
<sagede.applicationServer.authorization.policy>
  <allow name="Sage" service="*" action="*" />
  <allow
    name="Test"
    service="BelegService"
    action="http://Sagede/ApplicationServer/OfficeLine/Samples/BelegService/CreateBeleg"/>
</sagede.applicationServer.authorization.policy>
```

## Erläuterungen:

### Name

Name des (Sage-100-) Users

### Service

Name des Services

### Action

Name der Operation = Namespace des Service + „/“ + Name des Service + „/“ + Name der Operation

## Autorisierung der Applikation

Autorisierung auf Applikationsebene ist nur dann verfügbar, falls die gehostete Applikation dieses Feature unterstützt. Die Sage 100 unterstützt dieses Feature für SOAP-Services über ihr eigenes Berechtigungssystem.

Ist die Autorisierung durch die Applikation eingeschaltet, müssen dem Autorisierungssystem der Sage 100 die neuen, zu vergebenden Berechtigungen bekannt gemacht werden und die Benutzer, die die Services verwenden dürfen, berechtigt werden.

Das Ein- bzw. Ausschalten der Autorisierung durch die gehostete Applikation erfolgt in der Konfigurationsdatei auf Service-Ebene. Zum Einschalten muss für den Wert des Attributs **applicationAuthorization** = "true" eingetragen sein; zum Ausschalten "false".

## Ein- /Ausschalten der Autorisierung auf Applikationsebene in Konfiguration

```
<service
  name="BelegService"
  faultPromotion="None"
  applicationAuthorization="false"
  namespace="http://Sagede/ApplicationServer/OfficeLine/Samples">
```

...

</service>

## Sage 100 Funktionsberechtigungen für neue SOAP-Services erstellen

Wird das Programm „ASADMIN.exe“ mit dem Parameter „/a:OL“ ausgeführt, werden SQL-Skripte erzeugt, die Funktionsberechtigungen für den Service erzeugen bzw. wieder entfernen. Diese müssen auf dem SQL-Server auf allen Sage-100-Datenbanken ausgeführt werden, die von Services verwendet werden können und für die detaillierte Funktionsberechtigungen vergeben werden sollen.

Das folgende Beispiel zeigt ein solches generiertes Skript:

- Die erste Anweisung erzeugt eine Permission. Dabei ist Permissiongroup immer „Applikationsserver“. Der Name der Permission ist immer „svc“ + der Name des Service. [Type] muss immer 1 sein.
- Die zweite Anweisung erzeugt eine Funktion (Option) für diese Berechtigung.

Der Name der Funktion ist i.d.R. folgendermaßen aufgebaut:

Namespace des Service + „/“ + Name des Service + „/“ + Name der Operation.

Im Beispiel unten also:

### Namespace des Service:

http://sagede/ApplicationServer/OfficeLine/Samples

### Name des Service:

BelegService

### Operation (Funktion):

CreateBeleg

### Hinweise

Ist der Name der Funktion länger als 100 Zeichen so wird er auf 100 Zeichen gekürzt. Der gekürzte Name = „~“ + die letzten 99 Zeichen.

Der Wert für die „RightsMask“ muss immer auf 1 (lesen) gesetzt sein, da innerhalb des Applikationsservers immer dieses Recht überprüft wird.

### Beispiel:

```
INSERT INTO USysSecurityPermissions
```

```
(PermissionGroup,  
PermissionName,  
[Description],  
[Type])
```

```
VALUES
```

```
('Applikationsserver',  
'svcBelegService',  
'BelegService',  
1);
```

```
INSERT INTO USysSecurityOptions
```

```
(PermissionName,  
OptionName,  
[Description],  
RightsMask)
```

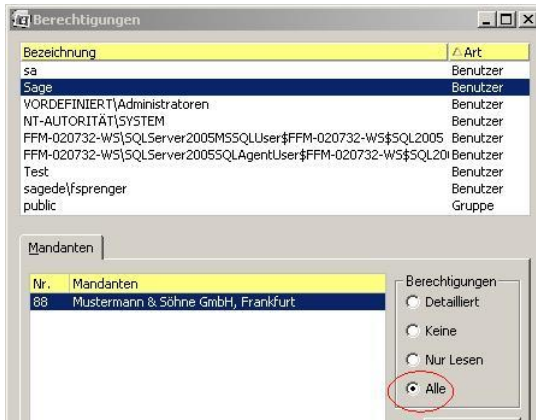
```
VALUES
```

```
('svcBelegService',  
'http://Sagede/ApplicationServer/OfficeLine/Samples/BelegService/CreateBeleg',  
'http://Sagede/ApplicationServer/OfficeLine/Samples/BelegService/CreateBeleg',  
1);
```

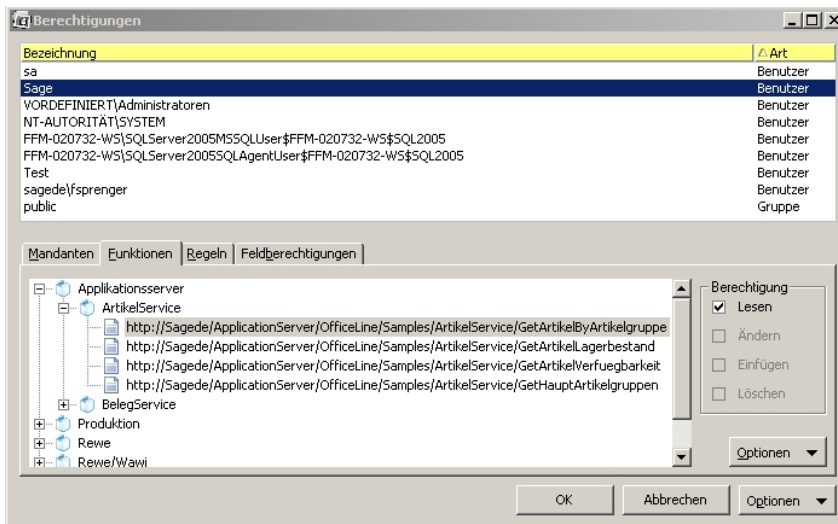
## Sage-100-Benutzer berechtigen

Berechtigung der Sage 100 Benutzer erfolgt über den „Sage 100 Administrator“. Zur Berechtigung eines Benutzers/ einer Benutzerin muss er/sie eines der folgenden Rechte besitzen:

### Das Recht „Alle“ bei den Mandantenberechtigungen



### Das Recht „Lesen“ bei den detaillierten Berechtigungen für die jeweiligen Funktion eines Services.



## SData Endpunkte konfigurieren

Die Konfiguration der SData Endpunkten erfordert:

- Einrichtung der SSL-Verbindung
- http Namespacereservierung
- Definition der Endpunkte in der Konfigurationsdatei „Sagede.ApplicationServer.SData.config“

Über ASADMIN.exe können diese einfach in einem Schritt ausgeführt werden:

### How to: Verwendung von ASADMIN.exe zur Konfiguration von SData Endpunkten

#### Aktuelle Konfiguration anzeigen:

```
ASADMIN.exe /command:sdataconfig /action:show
```

#### Aktuelle Konfiguration löschen:

```
ASADMIN.exe /command:sdataconfig /action:delete /all
```

Dieser Aufruf entfernt die für die SData Endpunkte konfigurierten Zertifikate, die SSL Bindungen an die SData Ports, die http Namespacereservierungen und die Endpunkte in der SData Konfigurationsdatei.

### Konfiguration erstellen:

**ASADMIN.exe /command:sdataconfig /action:create /profile:[profile]**

Erzeugt alle notwendigen Einstellungen entsprechend dem angegebenen Profil.

### Konfiguration wiederherstellen:

**ASADMIN.exe /command:sdataconfig /action:reset /profile:[profile]**

Entfernt die vorhandenen SData Einstellungen und erstellt sie neu entsprechend dem angegebenen Profil.

### Mögliche Angaben für [profile]:

- **runtime** – Profil für Laufzeitumgebung
  - Default-Wert, falls kein explizites Profil angegeben ist.
  - Lässt nur Https-Verbindungen zu. Verwendet den DNS-Name in der Adresse.
- **runtimeLocal** – Profil für lokale Laufzeitumgebung
  - Lässt nur Https-Verbindungen zu. Verwendet LocalHost in der Adresse.
- **development** – Profil für Entwicklungsumgebungen
  - Lässt Http- und Https-Verbindungen zu. Verwendet LocalHost in der Adresse für
  - Http-Verbindungen und den DNS-Namen für Https-Verbindungen. Auch minder-privilegierte
  - Accounts erhalten die Berechtigung die Http-Ports abzuhören.
- **developmentLocal** – Profil für lokale Entwicklungsumgebungen
  - Lässt Http- und Https-Verbindungen zu. Verwendet LocalHost in der Adresse für
  - Http- und Https-Verbindungen. Auch minder-privilegierte
  - Accounts erhalten die Berechtigung die Http-Ports abzuhören.

Alternativ können die SData-Endpunkte einzeln und spezifisch angelegt werden:

### Konfiguration eines Endpunktes erstellen:

**ASADMIN.exe /command:sdataconfig /action:create /port:[port]  
/binding:[webHttpBasic|webHttpsBasic|webHttpWindows|webHttpsWindows]**

Erzeugt einen SData-Endpunkt mit den angegebenen Eigenschaften.

Folgende Bindungen stehen zur Verfügung:

webHttpBasic – Web-Binding mit http-Transport und Basic-Authentication.

webHttpsBasic – Web-Binding mit https-Transport (SSL) und Basic-Authentication.

webHttpWindows – Web-Binding mit http-Transport und Windows-Authentication.

webHttpsWindows – Web-Binding mit https-Transport (SSL) und Windows-Authentication.

### Konfiguration eines Endpunktes löschen:

**ASADMIN.exe /command:sdataconfig /action:delete /port:[port]  
/binding:[webHttpBasic|webHttpsBasic|webHttpWindows|webHttpsWindows]**

Dieser Aufruf entfernt den SData-Endpunkt mit den angegebenen Eigenschaften.

# Secure Token Service konfigurieren

Die Konfiguration der Verwendung des Secure Token Service erfordert für den Server:

- Einrichtung der SSL-Verbindung
- http Namespacereservierung
- Erzeugung des Zertifikats für die Token-Signatur und Speicherung im Zertifikatsspeicher.
- Erzeugung der Konfigurationseinträge in „Sagede.ApplicationServer.Core.config“

Die Konfiguration der Verwendung des Secure Token Service erfordert für einen Client:

- ClientID und Secret vom STS-Server abfragen
- Zertifikat mit dem Public Key für die Überprüfung der Signatur im Zertifikatsspeicher hinterlegen
- Erzeugung der Konfigurationseinträge in „Sagede.ApplicationServer.Core.config“

Über ASADMIN.exe können diese einfach in einem Schritt ausgeführt werden:

## How to: Verwendung von ASADMIN.exe zur Konfiguration von STS-Server und Client bei Installation auf einer Maschine:

### Aktuelle Konfiguration anzeigen:

**ASADMIN.exe /command:stsconfig /action:show**

### Aktuelle Konfiguration löschen:

**ASADMIN.exe /command:stsconfig /action:delete /all**

Dieser Aufruf entfernt sämtliche STS-Konfiguration inklusive Zertifikate und SSL-Port Bindungen. Ohne Angabe von /all wird nur die STS-Server Konfiguration entfernt.

### Konfiguration erstellen:

**ASADMIN.exe /command:stsconfig /action:create /all**

Erzeugt alle notwendigen Einstellungen für STS-Server und Client. Ohne Angabe von /all wird nur der STS-Server konfiguriert..

### Konfiguration wiederherstellen:

**ASADMIN.exe /command:stsconfig /action:reset /all**

Entfernt die vorhandenen STS Einstellungen und erstellt sie neu.

### STS-Client Konfiguration exportieren:

**ASADMIN.exe /command:stsconfig /action:createregistration /usage:ApplicationServer /file:[Datei]**

Erzeugt die Einstellungen für einen STS-Client auf dem STS-Server und kopiert sie in die angegebene Datei.

### STS-Client Konfiguration importieren:

**ASADMIN.exe /command:stsconfig /action:importregistration /file:[Datei]**

Importiert die Client-Konfiguration aus einer exportierten Datei auf einem Applikationsserver.

## Client-seitiges Loadbalancing für die Sage 100

Wenn ein Applikationsserver gestartet wird, teilt er dies dem Treiber der Applikation mit. Zusätzlich werden die vorhandenen SData Endpunkte übergeben. Innerhalb des Servers kann in dessen Konfiguration hinterlegt werden, für welchen Zweck dieser Server verwendet werden kann (z.B. als Server für Sage 100 Clients). Diese Informationen werden im einem Integer-Wert (Capability) angegeben. Der Sage 100 Treiber des Applikationsservers nimmt beim Start des AS die Endpunkte und die Default-Capability entgegen und schreibt Informationen über diese Applikationsserver Instanz

in die Remote-Registry in den Zweig *Application Server* unterhalb des Sage 100 Zweiges. Über diese Information finden Clients ihre Endpunkte.

Der Capability-Wert wird in der Konfigurationsdatei

Sagede.ApplicationServer.Core.config eingetragen

(„server/applicationEnvironment/@capability“). Ist kein Wert oder "-1" eingetragen, trägt der Treiber der Sage 100 "3" (= 1 + 2) als Capability Wert in die Remote Registry ein.

Für die Sage 100 wird folgende Regel für die Capability festgelegt:

1 = Sage 100 Clients können diese AS Instanz verwenden.

2 = Auf dieser Instanz des AS liegen die Sage 100 Metadaten Dateien. Entwicklung via AppDesigner erfolgt auf dieser Instanz.

4 = Reserviert für CRM Serverintegration

8 = Reserviert für Reporting Server

Wird ein Applicationserver installiert, der nicht am Sage 100 Loadbalancing teilnehmen soll, muss manuell capability="0" eingetragen werden. Wird ein weiterer Applikationsserver installiert, der am Sage 100 Loadbalancing teilnimmt, aber nicht als Entwicklungsrechner für den AppDesigner dient, muss manuell capability="1" eingetragen werden.

Sofern mehrere Applikationsserver zur Verfügung stehen, muss es einen Algorithmus geben, über den die Clients auf die vorhandenen Server verteilt werden. Idealerweise kommt ein User immer wieder auf den gleichen Applikationsserver, da die Isolationsprozesse unter einem Benutzerkontext laufen. Damit ist die Wiederverwendbarkeit eines Prozesses wahrscheinlicher. Dies wird folgendermaßen erreicht:

Der Client der Sage 100 liest die Informationen aus der Remote-Registry. Das Default-Loadbalancing erfolgt über den Namen des Benutzers. Der Index der zu verwendenden Applikationsserver Instanz ergibt sich dann aus [Hash des Benutzernamens] modulo [Anzahl Applikationsserver Instanzen].

## Blobstorage Server konfigurieren

Die Konfiguration der Verwendung des Blobstorage Servers erfordert:

- Einrichtung der Endpunkte inklusive SSL-Verbindung
- http Namespacereservierung
- ClientID und Secret vom STS-Server abfragen
- Zertifikat mit dem Public Key für die Überprüfung der Signatur im Zertifikatsspeicher hinterlegen

Über BSADMIN.exe können diese einfach in einem Schritt ausgeführt werden:

### How to: Verwendung von BSADMIN.exe zur Konfiguration des Blobstorage Servers:

#### Endpunkte erzeugen und Speicherort festlegen:

```
BSAdmin /action:create /storageroot:c:\storage
```

Erzeugt die Endpunkte des Servers und legt den Speicherort für die Dateien fest.

#### Aktuelle Konfiguration löschen:

```
BSADMIN.exe /action:delete /all
```

Dieser Aufruf entfernt sämtliche Blobstorage-Konfiguration inklusive Zertifikate und SSL-Port Bindungen.

#### Blobstorage für die Anwendung registrieren:

```
BSADMIN.exe /command:application /action:add /appid:ol62 /server:[Mehrbenutzerdienst]  
[/pwd:[passwort]]
```

Registriert den Blobstorage Server für die Anwendung via Eintrag in die RemoteRegistry des angegebenen Mehrbenutzerdienstes..

### **Blobstorage STS-Client Konfiguration einrichten:**

**BSADMIN.exe /command:importsts /fromfile:[Datei]**

Erstellt die Anbindung des Blobstorage Server an einen STS (siehe oben) unter verwendung einer STS-Client Konfigurationsdatei.

### **Liveupdate ausführen:**

**BSADMIN.exe /command:liveupdate**

Startet das Liveupdate für den Blobstorage Server.

# Administration und Diagnose

Die Administration des Applikationsservers erfolgt über den Basis-Konfigurationsdialog, das MMC-Snapin und Einstellungen in der zugeordneten Konfigurationsdatei. Zur Diagnose des Zustands des Applikationsservers stehen Performance Counter und die Auswertung der Log-Datei zur Verfügung.

## » Abschnitte dieses Kapitels

- Liveupdate
- Tracing
- Performance-Counter
- Servermanager

## Liveupdate

Das Liveupdate des Applikationsservers wird aus dem Startmenü aufgerufen. Vor dem Liveupdate muss der Server heruntergefahren werden (Dienst beenden).

## Konfiguration

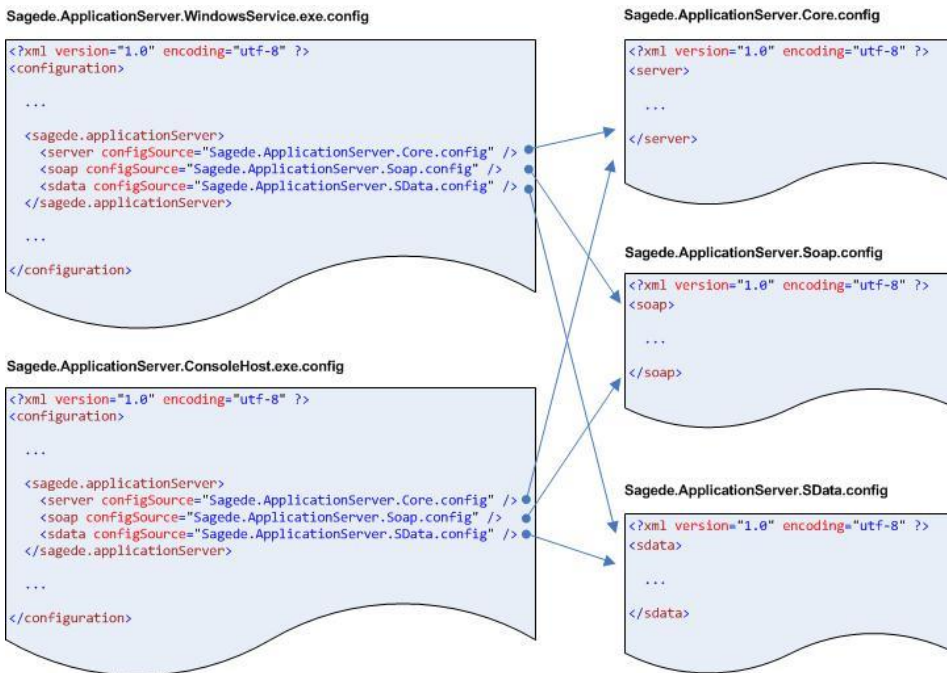
Der Applikationsserver verwendet den Standard-.NET Mechanismus zur Verwaltung von Konfigurationseinstellungen. Die Konfigurationsdateien sind:

- `Sagede.ApplicationServer.WindowsService.exe.config` für den Windows Dienst
- `Sagede.ApplicationServer.ConsoleHost.exe.config` für den Consolen Host

In ihnen sind Verweise auf gemeinsam verwendete Konfigurationsdateien und spezifische Konfigurationsangaben für das Logging enthalten.

- `Sagede.ApplicationServer.Core.config` für die grundsätzliche Konfiguration des Servers
- `Sagede.ApplicationServer.Soap.config` für die Konfiguration der SOAP-Dienste
- `Sagede.ApplicationServer.SData.config` für die Konfiguration der SData-Dienste
- `Sagede.BlobStorageServer.exe.config` für die Konfiguration des Blobstorage Servers





## Achtung

Änderungen an den Konfigurationsdateien werden erst nach einem Neustart des jeweiligen Applikationsserver Prozesses wirksam! Änderungen in den spezifischen Konfigurationsdateien wirken sich auf beide Hosts aus.

Zusätzlich gibt es noch eine Konfigurationsdatei für den Isolations-Host für die Isolation via Prozessen: Sagede.ApplicationServer.IsolationProcess.exe.config.

## Grundlegender Aufbau einer Host-Konfigurationsdatei

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

<configSections>
  <sectionGroup
    name="sagede.applicationServer"
    type="Sagede.ApplicationServer.Configuration.ApplicationServer
      SectionGroup, Sagede.ApplicationServer">
    <section name="server"
      type="Sagede.ApplicationServer.Configuration.ServerSection,
        Sagede.ApplicationServer" />
    <section name="soap"
      type="Sagede.ApplicationServer.Configuration.SoapSection,
        Sagede.ApplicationServer" />
    <section name="sdata"
      type="Sagede.ApplicationServer.Configuration.SDataSection,
        Sagede.ApplicationServer" />
  </sectionGroup>
</configSections>

...

<sagede-applicationServer>
  <server configSource="Sagede.ApplicationServer.Core.config" />
  <soap configSource="Sagede.ApplicationServer.Soap.config" />
  <sdata configSource="Sagede.ApplicationServer.SData.config" />
</sagede-applicationServer>
```

...

```
<system.diagnostics> ... </system.diagnostics>
```

```
</configuration>
```

## Erläuterungen zum grundlegenden Aufbau einer Konfigurationsdatei:

```
<configSections>
```

Innerhalb des `<configSections>`-Tags müssen die `SectionGroup` ("`sagede.applicationServer`") und die enthaltenen Child-Sections ("`server`", "`soap`" und "`sdata`") mit den entsprechenden Handler definiert werden.

```
<sagede.applicationServer>
```

Das `<sagede.applicationServer>`-Tag enthält die Verweise und die eigentlichen, spezifischen Konfigurationsdateien des Applikationsservers.

```
<system.diagnostics>
```

In `<system.diagnostics>` befinden sich Einstellungen, die das Logging und Tracing steuern.

## Einstellungen in Sagede.ApplicationServer.Core.config

Diese Konfigurationsdatei enthält Einstellungen für die Infrastruktur und Default-Einstellungen, die in den Service-Definitionen überschrieben werden können.

```
<?xml version="1.0" encoding="utf-8" ?>  
<server>
```

```
<isolationServer> ... </isolationServer>
```

```
<applicationEnvironment  
  applicationContextFactory="Sagede.ApplicationServer.  
    OfficeLine.ApplicationDriver,  
    Sagede.ApplicationServer30.OfficeLine.Driver"  
  applicationBinaryPath="c:\Programme\Sage\Sage 100\8.0" />
```

```
<security> ... </security >
```

```
<quota profile = "Intranet"> ... </quota>
```

```
<diagnostics performanceCounter="true" />
```

```
</server>
```

### Erläuterungen zu `<server>`:

```
<isolationServer>
```

`<isolationServer>` enthält allgemeine Konfigurationseinstellungen für den Applikationsserver Kern.

```
<asyncIsolationServer>
```

`<asyncIsolationServer>` enthält allgemeine Konfigurationseinstellungen für Server-seitig asynchrone SData Anfragen gegen den Applikationsserver Kern.

```
<applicationEnvironment>
```

`<applicationEnvironment>` enthält unter `@applicationContextFactory` den Typnamen der Implementierung von „IApplicationDriver“ der gehosteten Applikation ~ Treiberimplementierung der gehosteten Applikation. Der Pfad zu den Binärdateien der gehosteten Applikation muss in `@applicationBinaryPath` enthalten sein.

`@capability` (default = -1) steuert das Client-seitige Loadbalancing (siehe „Client-seitiges Loadbalancing“).

`<security>`

`<security>` enthält die Definition des Autorisierungsmanagers für die SOAP-Frontend-Autorisierung. Weiterhin kann hier die Security-Policy konfiguriert werden.

`<quota>`

`<quota>` enthält die Defaulteinstellungen für Transport Quotas. Falls in einer Service-Konfiguration (SOAP und/oder SData) nicht explizit gesetzt, so werden diese Einstellungen verwendet. Details s.u.

`<diagnostics>`

Über `<diagnostics>` können die Applikationsserver spezifischen PerformanceCounter aus- bzw. eingeschaltet werden.

## Einstellungen in `<server>/<isolationServer>`

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<configuration>
```

```
<isolationServer
  maxActiveServiceDomains="0"
  messageBufferTimeout="00:00:05"
  serviceDomainSynchronizationLockTimeout="00:00:01.500"
  synchronizeContextSwitch="false"
  synchronizeContextSwitchTimeout="00:00:01.000">
  maxUseServiceDomains="0"
  maxServiceDomainMemorySize="0"
```

```
<lifetimeManager
  serviceDomainPollTime="00:01:00" />
```

```
<serviceDomain>
  lifeTimeLeaseTime="01:00:00"
  lifetimeRenewOnCalltime ="01:00:00"
  isolationModel ="process"
  poolInitialSize ="10"
  poolMinSize ="2"
  poolMaxSize ="20"
  contextQuotaMax ="2"
  contextQuotaTimeout ="00:00:10"
  maxElasticCount ="0" />
```

```
</isolationServer>
```

```
...
```

```
</configuration>
```

## Erläuterungen zu `<server>/<isolationServer>`:

### `<isolationServer>`

Unter `<isolationServer>` stehen alle Einstellungen des eigentlichen Isolation-Servers:

- **@maxActiveServiceDomains**  
definiert die maximale Anzahl von möglichen parallelen Anfragen. Ist die maximale Anzahl erreicht, müssen weitere Request auf die Freigabe vorhandener aktiver Service-Domains warten. Der Einstellung „0“ bedeutet, dass keine Quotierung erfolgt. Bei einem negativen Wert wird eine Begrenzung auf [Wert] \* [Anzahl Cores] vorgenommen. D.h. bei einer 4-Core Maschine bedeutet „-3“, dass maximal 12 Requests parallel erlaubt sind.
- **@messageBufferTimeout**  
Maximale Zeit, die gewartet wird, falls die maximale Anzahl von parallelen Request überschritten wurde. Nach dieser Zeit wird ein Fehler ausgelöst, der anzeigt, dass der Server ausgelastet ist.
- **@serviceDomainSynchronizationLockTimeout**  
definiert einen Zeitraum der angibt, welches Timeout beim Setzen eine Sperre in der internen Verwaltungsstruktur verwendet wird. Nach dieser Zeit wird ein Fehler ausgelöst, der anzeigt, dass der Server ausgelastet ist.
- **@synchronizeContextSwitch**  
gibt an, ob der Kontext-Wechsel einer ServiceDomain serialisiert werden muss.
- **@synchronizeContextSwitchTimeout**  
definiert einen Zeitraum der angibt, wie lange der Server wartet, um einen Kontext-Wechsel seriell durchzuführen.
- **@maxUserServiceDomains**  
nach dieser Anzahl von Anfragen gegen eine ServiceDomain wird die ServiceDomain entladen und neu gestartet („0“ = Keine Überprüfung).
- **@maxServiceDomainMemorySize**  
überschreitet der Speicherverbrauch einer ServiceDomain diesen Wert (in KB), wird die ServiceDomain entladen und neu gestartet („0“ = Keine Überprüfung).

### `<isolationServer>/<lifetimeManager>`

`<isolationServer>/<lifetimeManager>` steuert das Verhalten der Komponente, die die Lebensdauer einer Service-Domain überwacht:

- **@serviceDomainPollTime**  
setzt das Zeitintervall, nach dem jeweils überprüft wird, ob eine Service-Domain abgeräumt werden kann. Ein abgeräumte ServiceDomain wird deinitialisiert und den Pool des jeweiligen Service geschoben. Falls der Pool bereits voll ist, wird die ServiceDomain zerstört.

`<isolationServer>/<serviceDomain>`

`<isolationServer>/<serviceDomain>` definiert die Einstellungen von Service-Domains innerhalb des Isolation-Servers (diese Einstellungen können per Service überschrieben werden):

- **@lifeTimeLeaseTime**  
Initiale Lebensdauer einer Service-Domain.
- **@lifetimeRenewOnCalltime**  
Nachdem die Initiale Lebensdauer abgelaufen ist, die erneuerte Lebensdauer der Service-Domain.
- **@isolationModel**  
Der zu verwendene Isolationsmechanismus. Möglich sind „process“ und „appdomain“. Per Default werden Prozesse („process“) zur Isolation verwendet.
- **@poolInitialSize**  
Gibt die Anzahl von ServiceDomains an, die beim Start des Servers per Service erzeugt und vorgeladen werden sollen.
- **@poolMaxSize**  
Gibt die Anzahl von ServiceDomains an, die maximal im Pool eines Services liegen dürfen.
- **@poolMinSize**  
Gibt die Anzahl von ServiceDomains an, die mindestens im Pool eines Services vorhanden sein sollen. Wird dieser Wert bei der Entnahme einer ServiceDomain aus dem Pool unterschritten und kann eine weitere ServiceDomain erzeugt werden, so wird im Hintergrund eine neue ServiceDomain vorgeladen und in den Pool gelegt.
- **@maxElasticCount**  
Beschränkt die maximale Anzahl von ServiceDomains per Service. Diese setzt sich aus den ServiceDomains der Verwaltungsstruktur plus den ServiceDomains im Pool zusammen. Da der Ezeugungprozess für ServiceDomains für den Pool asynchron funktioniert, kann die tatsächliche Anzahl bei der Beschränkung um 1-2 nach oben abweichen. „0“ bedeutet, daß keine Beschränkung existiert.
- **@contextQuotaMax**  
Definiert die maximale Anzahl von Requests, die bei gleichem Kontext (i.d.R. gleicher Client) parallel ausgeführt werden können. Übersteigt die aktuelle Anzahl diesen Wert, so wird mit der Ausführung gewartet, bis die aktuellen Request für diesen Kontext abgeschlossen sind oder die Zeit, die in **@contextQuotaTimeout** definiert ist, abgelaufen ist. „0“ bedeutet, daß keine Beschränkung existiert. Eine Quotierung über diese Einstellung ermöglicht eine fairer Verteilung der Ressourcen über die anfragenden Clients.
- **@contextQuotaTimeout**  
Definiert den Zeitraum, der gewartet wird, falls die maximale Anzahl von Requests, die bei gleichem Kontext (i.d.R. gleicher Client) überschritten ist. In diesem Fall wird ein Fehler ausgelöst, der anzeigt, daß der Server ausgelastet ist.

## Einstellungen in `<server>/<asynclIsolationServer>`

Die möglichen Einstellungen entsprechen denen, für `<server>/<isolationServer>` Im folgenden sind nur die Abweichungen beschrieben.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
```

```
  <asynclIsolationServer
  ...
  asyncQueuePollTime="00:00:01.000"
  ...
</asynclIsolationServer>
...
</configuration>
```

**@asyncQueuePollTime**

Intervall nach dem das System die Warteschlange nach neuen Anfragen abfragt.

## Einstellungen in `<server>/<security>`

```
<?xml version="1.0" encoding="utf-8" ?>
<server>
  ...
  <security>
    <authorizationManager
      provider="Sagede.ApplicationServer.Security.
XmlFileServiceActionAuthorizationDataProvider,
      Sagede.ApplicationServer" connection="Sagede.ApplicationServer.Authorization.xml"
      enabled="true"
    />
    <securityPolicy
      enabled="true"
      maxRetries="3"
      waitOnLockTime="00:10:00"
    />
    <stsClient
      enabled="true"
      host="https://mystsserver:5466/"
      clientId="617e073c-5877-47d0-949a-afb78702671e"
      clientsecret="b7bc4b97-8733-48e6-a00a-dbd5687d22d2"
    />
    <stsServer
      enabled="true"
      host="https://mystsserver:5466/"
      certificateThumbprint="0D3904FC60DD59D95BA77F3C828469C6079FADCE"
      tokenExpiration="600"
    />
  </security>
  ...
</server>
```

## Erläuterungen zu `<server>/<security>`:

`<security>/<authorizationManager>`

Unter `<security>/<authorizationManager>` ist die Autorisierung der Frontend Facade für den gesamten Server konfiguriert. Ist dieses Tag nicht vorhanden, so ist diese Autorisierung deaktiviert. SData Services verwenden keine Frontend-Autorisierung.

- **@provider**  
@provider gibt den Typenamen des Datenproviders zurück, über den die Autorisierungsdaten gelesen werden können.
- **@connection**  
In @connection werden die Verbindungsinformationen zur Datenquelle angegeben.
- **@enabled**  
Via @enabled kann diese Autorisierung an- bzw. ausgeschaltet werden.

`<security>/<securityPolicy>`

Unter `<security>/<securityPolicy>` ist angegeben, wie sich der Server bei fehlerhafter Anmeldung verhält.

- **@enabled**  
Via @enabled kann diese Policy an- bzw. ausgeschaltet werden.
- **@maxRetries**  
Maximale Anzahl von Fehlversuchen, bevor der Aufrufer gesperrt wird.
- **@waitOnLockTime**  
Der Zeitraum, den der Aufrufer nach @maxRetries Fehlversuchen gesperrt wird.

## <security>/<stsClient>

Unter <security>/<stsClient> ist angegeben, gegen welchen STS-Server die Authentisierung von SData-Anfragen durchgeführt werden soll.

- **@host**  
Die Basisadresse des STS-Servers.
- **@clientid**  
Die ID dieses Clients beim STS.
- **@clientSecret**  
Das Secret, mit dem sich dieser Client beim STS-Server authentisiert.

## <security>/<stsServer>

Unter <security>/<stsServer> wird der interne STS-Server konfiguriert.

- **@enabled**  
Via **@enabled** kann dieser Server an- bzw. ausgeschaltet werden.
- **@host**  
Die Basisadresse des STS-Servers.
- **@certificateThumbprint**  
Der Fingerabdruck des Zertifikats (in Computer/Eigene Zertifikate), das für die Signierung des Tokens verwendet werden soll.
- **@tokenExpiration**  
Der Gültigkeitszeitraum des ausgestellten Tokens in Sekunden.

## Einstellungen in <server>/<quota>

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<server>
```

```
...
```

```
<quota
```

```
  profile = "Intranet"
```

```
  maxArrayLength = "[int]"
```

```
  maxBytesPerRead = "[int]"
```

```
  maxDepth = "[int]"
```

```
  maxNameTableCharCount = "[int]"
```

```
  maxStringContentLength = "[int]"
```

```
  receiveTimeout = "[Timespan]"
```

```
  sendTimeout = "[Timespan]"
```

```
  maxReceivedMessageSize = "[int]"
```

```
  maxConcurrentCalls = "[int]"
```

```
  maxConcurrentSessions = "[int]"
```

```
  maxItemsInObjectGraph = "[int]"
```

```
>
```

```
</server>
```

## Erläuterungen zu <server>/<quota>

### @profile

Ist **@profile** nicht angegeben oder = "Default", so werden die Default-WCF Einstellungen verwendet. Ist **@profile** = "Intranet", so werden alle Einstellungen auf ihren größtmöglichen Wert gesetzt. Die folgenden Attribute überschreiben, soweit angegeben, die Default-Werte, die über das **@profile** gesetzt sind.

## Einstellungen in Sage.de.ApplicationServer.Soap.config

Diese Konfigurationsdatei enthält Einstellungen für die veröffentlichten Soap-Dienste.

```

<?xml version="1.0" encoding="utf-8" ?>
<soap>

  <endpointConfiguration> ... </endpointConfiguration>

  <serviceCertificate> ... </serviceCertificate>

  <services> ... </services>

</soap>

```

## Erläuterungen zu **<soap>**:

**<endpointConfiguration>**

**<endpointConfiguration>** enthält Defaulteinstellungen für Services. Falls in einer Service-Konfiguration nicht explizit gesetzt, so werden diese Einstellungen verwendet.

**<serviceCertificate>**

Unter **<serviceCertificate>** ist angegeben, wie der Server das Zertifikat, das für den sicheren tcp-Transport erforderlich ist, gefunden werden kann.

**<services>**

Unterhalb von **<services>** sind alle öffentlichen SOAP-Services definiert.

## Einstellungen in **<soap>/<endpointConfiguration>**

```

<?xml version="1.0" encoding="utf-8" ?>
<soap>

  <endpointConfiguration>

    <metadataPublishing httpEnabled="true" httpsEnabled="true"/>

    <baseAddresses>
      <baseAddress url="http://localhost:7777/myservices"/>
      <baseAddress url="net.tcp://localhost:7778/myservices"/>
    </baseAddresses>

    <endpoints>
      <endpoint
        address="basic"
        binding="basicHttpBinding"
      />
      <endpoint
        address="bin"
        binding="binaryHttpBinding"
      />
    </endpoints>

  </endpointConfiguration>

  ...

</soap>

```



## Erläuterungen zu `<soap>/<endpointConfiguration>`

### `<endpointConfiguration>`

Unter `<soap>/<endpointConfiguration>` ist die Defaulteinstellung der SOAP-Serviceendpunkte definiert. Falls keine abweichenden Angaben in der Konfiguration eines Services vorgenommen wurden, so wird diese verwendet.

### `<endpointConfiguration>/<metadataPublishing>`

`<metadataPublishing>` schaltet den MEX-Endpunkt via http bzw. https ein bzw. aus:

- `httpEnabled="[true | false]"`
- `httpsEnabled="[true | false]"`

### `<endpointConfiguration>/<baseAddresses>/<baseAddress[@url]>`

`<baseAddresses>/<baseAddress[@url]>` setzt die protokollspezifische Basisadresse, unter der der Service zu erreichen ist. Ein http-Endpunkt verwendet z.B. die Basisadresse, deren `url` mit „http“ beginnt. Die finale Adresse eines Services setzt sich aus Basisadresse, dem Namen des Services und `address` des Endpunktes zusammen. Die zusammengesetzte Adresse eines Services muss eindeutig sein.

### `<endpointConfiguration>/<endpoints>`

Unter `<endpoints>` sind alle Endpunkte mit ihrem Adressanteil und ihrem Binding definiert. Folgende Einstellung für `binding` sind möglich:

Wert	Transport	Encryption	Encoding	Modus
<code>basicHttpBinding</code>	http	-	XML	Buffered
<code>binaryHttpBinding</code>	http	-	Binär	Buffered
<code>basicHttpsBinding</code>	https	SSL	XML	Buffered
<code>binaryHttpsBinding</code>	https	SSL	Binär	Buffered
<code>netTcpBinding</code>	tcp/ip	-	Binär	Buffered
<code>netTcpSecBinding</code>	tcp/ipsec	SSL	Binär	Buffered
<code>basicHttpStreamedBinding</code>	http	-	XML	Streamed
<code>binaryHttpStreamedBinding</code>	http	-	Binär	Streamed
<code>basicHttpsStreamedBinding</code>	https	SSL	XML	Streamed
<code>binaryHttpsStreamedBinding</code>	https	SSL	Binär	Streamed
<code>netTcpStreamedBinding</code>	tcp/ip	-	Binär	Streamed
<code>netTcpSecStreamedBinding</code>	tcp/ipsec	SSL	Binär	Streamed
<code>netNamedPipeBinding</code>	ipc	-	Binär	Buffered
<code>netNamedPipeStreamedBinding</code>	ipc	-	Binär	Streamed

## Einstellungen in `<soap>/<serviceCertificate>`

```
<?xml version="1.0" encoding="utf-8" ?>
<soap>
  ...
  <serviceCertificate
    storeLocation="LocalMachine"
    storeName="My"
    thumbprint="ac80020f6727b6f69280560ef541b3a147f04bf6"
  />
  ...
</soap>
```

## Erläuterungen zu `<soap>/<serviceCertificate>`:

Unter `<soap>/<serviceCertificate>` ist angegeben, wie der Server das Zertifikat, das für den sicheren tcp-Transport erforderlich ist, gefunden werden kann.

- **@storeLocation**  
Gültige Werte für @storeLocation sind:
  - ["LocalMachine"|"CurrentUser"]
- **@storeName**  
Mögliche Angaben für @storeName sind:
  - ["AddressBook" | "AuthRoot" | "CertificateAuthority" | "Disallowed" | "My" | "TrustedPeople" | "TrustedPublisher"]
- **@thumbprint**  
Unter @thumbprint ist der Fingerabdruck des Zertifikats ohne Leerzeichen zu hinterlegen.

## Einstellungen in `<soap>/<services>`:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap>
  ...
  <services>

    <service
      name="CoreService"
      faultPromotion="All"
      namespace="http://sage.de/OL/services/V1"
      applicationAuthorization="true">

      <serviceDomain> ... </serviceDomain/>
      <endpointConfiguration> ... </endpointConfiguration/>
      <quota> ... </quota/>

      <contract type="[typeName] />
      <implementation type="[typeName] />

    </service>

  </services>

  ...
</soap>
```

## Erläuterungen zu `<soap>/<services>`:

```
<services>
```

Unterhalb von `<services>` sind alle öffentlichen SOAP-Services definiert. Für jeden Service existiert ein `<service>` Eintrag. Innerhalb eines Services können optional folgende Defaulteinstellungen überschrieben

werden (Beschreibung siehe oben):  
<serviceDomain>, <endpointConfiguration>, <quota>

<service[@name]>

Definiert den Namen des Services. Ist ein Bestandteil der finalen Adresse des Services.

<service[@faultPromotion]>

Definiert, welche Fehler zum Client transportiert werden sollen.

Mögliche Wert sind:

Wert	Beschreibung
All	Es werden alle Fehler in SOAP-Faults umgewandelt und an den Client übertragen.
None	Es wird immer eine Standardfehlermeldung zurückgegeben. Eine Ausnahme bilden „Zugriff verweigert“-Faults. Diese werden immer zum Client zurückgeschickt.
Service	Wie „None“. Zusätzlich werden alle Fehler an den Client übertragen, die aufgrund einer „ApplicationServiceException“ aufgetreten sind.

<service[@namespace]>

Setzt den eindeutigen Namespace des Service – falls nicht angegeben, wird intern ein Default-NS verwendet.

<service[@applicationAuthorization]>

Gibt an, ob Autorisierung auf Applikationsebene verwendet werden soll (default = „true“).

<contract[@type]>

Liefert den vollen Typnamen des Serviceinterfaces. Format: [MyNamespace.IServiceInterface], [MyAssemblyName]. Der Name der Assembly ist ohne Dateieindung anzugeben.

<implementation[@type]>

Liefert den vollen Typnamen der das Serviceinterface implementierenden Klasse. Format s.o.

## Einstellungen in Sage.ApplicationServer.SData.config

Diese Konfigurationsdatei enthält Einstellungen für die Konfiguration der SData-Endpunkte.

```
<?xml version="1.0" encoding="utf-8" ?>
<sdata>

  <trackingStorage> ... </trackingStorage>

  <sdataService faultDetails="false">
    ....<serviceDomain>..</serviceDomain>
    ....<quota>..</quota>
    <sdataEndpoints>..<sdataEndpoints>
  ..</sdataService>

</sdata>
```

### Erläuterungen zu <sdata>:

<trackingStorage>

<trackingStorage> enthält Konfigurationseinstellungen für den Storage für asynchrone SDataServices.

<sDataService>

<sDataService> definiert die SData-Endpunkte und optionale Überschreibungen des Default <quota> Elements und des Default <serviceDomain> Elements.

<sDataService>[@faultDetails]

@faultDetails gibt an, ob im Fehlerfall detaillierte Fehlerinformation an den Client zurückgeliefert werden sollen.

### Einstellungen in <sdata>/<trackingStorage>:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<sdata>
```

```
<trackingStorage>
  dbTimeout="20000"
  pollTime="04:00:00"
  lifeTime="72:00:00"
  enabled="true"
/>
```

```
</sdata>
```

### Erläuterungen zu <sdata>/<trackingStorage>:

@dbTimeout

Setzt einen Wert für den Timeout, nachdem keine weiteren Schreibzugriff bei gesperrter Datenbank erfolgen sollen. Der Wert ist in Millisekunden anzugeben.

@pollTime

Setzt das Zeitintervall, nach dem jeweils überprüft wird, ob Einträge im Trackingstorage abgelaufen sind und deshalb entfernt werden.

@lifeTime

Setzt das Lebensdauer eines Eintrags im TrackingStorage. Ist diese Zeit abgelaufen, so wird der Eintrag entfernt.

@enabled

Gibt an, ob die Überprüfung der Lebensdauer ein- bzw. ausgeschaltet ist.

### Einstellungen in <sdata>/<sDataService>/<sdataEndpoints>:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<sdata>
```

```
<sDataService>
```

```
<sdataEndpoints>
```

```
<sdataEndpoint
  address="[protocol]://[host]:[port]/sdata"
  binding="[binding]">
```

```
</sdataEndpoints>
```

```
..</sDataService>
```

```
</sdata>
```

## Erläuterungen zu `<sdata>/<sdataService>/<sdataEndpoints>`:

Die in `<sdataEndpoints>` enthaltenen `<sdataEndpoint>` Elemente definieren die öffentlichen SData-Endpunkte des Servers. Die Default-SData Endpunkte können via ASADMIN.exe generiert werden.

`<sdataEndpoint>[@address]`

Definiert die Adresse des Endpunktes. Eine Adresse setzt sich folgendermaßen zusammen:

- Protocol: „http“ oder „https“
- Host: DNS-Name des Servers
- Port: Der Port des Endpunktes
- Schlüsselwort „sdata“

Beispiel: „https://MeinHost.Acme.de:5493/sdata“

`<sdataEndpoint>[@binding]`

Definiert die Art der Bindung des Endpunktes. Folgende Einstellung für `binding` sind möglich:

Wert	Transport	Encryption	Authentication
webHttpBasic	http	-	Basic
webHttpWindows	http	-	Windows
webHttpsBasic	https	SSL	Basic
webHttpsWindows	https	SSL	Windows

Es ist zu beachten, daß die Angabe des Protokolls in der Adresse mit dem Binding korreliert. Die http-Endpunkte dienen nur zur Entwicklung von SData-Services. In Produktiv-Umgebungen dürfen nur https-Endpunkte veröffentlicht werden.

## Einstellungen in `Sagede.BlobStorageServer.exe.config`

Diese Konfigurationsdatei enthält Einstellungen für den Blobstorage-Dienst.

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<configuration>
```

```
<configSections>
```

```
<section name="BlobStorage" type="Sagede.BlobStorageServer...">
```

```
</configSections>
```

```
<BlobStorage StorageRoot="c:\DataFolder">
```

```
<Endpoints> ... </Endpoints>
```

```
<stsClient> ... </stsClient>
```

```
<MultiuserServer name="\\" RegistryKey="Software\Sage\Office Line8.0"/>
```

```
</BlobStorage>
```

```
</configuration>
```

### Erläuterungen zu `<BlobStorage>`:

`@StorageRoot`

Enthält den Pfad, unter dem der Blobstorage seine Dateien speichert..

`<Endpoints>`

Unter `<Endpoints>` sind alle Endpunkte angegeben, über die der Blobstorage zu erreichen ist..

`<MultiuserServer>`

## Tracing

Tracing kann über zwei unterschiedliche Verfahren und Technologien erfolgen. Das erste Verfahren verwendet die Standard System.Diagnostics Tracing Technologie. Das zweite Verfahren verwendet die Ereignisablaufverfolgung für Windows (ETW).

### Standard System.Diagnostics Tracing

Informationen werden in die Log-Datei des Applikationsservers, Sage.de.ApplicationServer.Log, geschrieben. Diese liegt im Root-Verzeichnis des Applikationsservers. Der Server protokolliert fortlaufend die wichtigsten Aktionen, wie z. B. Serviceaufrufe und Fehlermeldungen. Bis zu einem gewissen Grad kann man steuern welche und wieviele Informationen protokolliert werden. Der Grund ist, dass zu ausführliches Logging Performance kosten würde und zudem die Logdatei zu schnell zu groß wird. Deshalb gibt es verschiedene Loglevel, diese werden über folgende Konfigurationsdateien des Applikationsservers gesetzt:

- Sage.de.ApplicationServer.WindowsService.exe.config
- Sage.de.ApplicationServer.ConsoleHost.exe.config
- Sage.de.ApplicationServer.IsolationProcess.exe.config

Dort gibt es jeweils einen Abschnitt „System.Diagnostics“, in welchem verschiedene Parameter gesetzt werden können.

```
<system.diagnostics>
<trace autoflush="true" />
<switches>
  <add name="ServerSwitch" value="Information"/>
</switches>
<sources>
  <source name="Sagede.Application.Server" switchName="ServerSwitch"...>
    <listeners>
      <add name="File Logger"
        type="Sagede.ApplicationServer.Diagnostics.DefaultFileTraceListener,
          Sage.de.ApplicationServer" />
      <add name="Console Logger"
        type="Sagede.ApplicationServer.Diagnostics.
          DefaultConsoleTraceListener, Sage.de.ApplicationServer" />
      <remove name="Default" />
    </listeners>
  </source>
</sources>
</system.diagnostics>
```

### Hinweis

**Falls der Isolationsmodus „process“ eingestellt ist (default), wird das Tracing von Meldungen aus den Prozessen über die Konfigurationsdatei „Sagede.ApplicationServer.IsolationProcess.exe.config“ gesteuert. Soll das LogLevel für alle Meldungen geändert werden, so muss dies in der Konfigurationsdatei des Hosts und in der Konfigurationsdatei der Prozesse geändert werden.**

## Erläuterungen zum Abschnitt „System.Diagnostics“

### <ServerSwitch>

Das LogLevel setzt man im Attribut <ServerSwitch> auf einen der folgenden verfügbaren Werte:

- Off  
Schaltet das Logging ab.
- Critical  
Bei „Critical“ werden nur Informationen zu kritischen Fehlern protokolliert.
- Error  
Weil die Levels additiv sind, deshalb enthält „Error“ auch Critical.
- Warning  
Weil die Levels additiv sind, deshalb enthält „Warning“ auch Critical und Error.
- Information  
Weil die Levels additiv sind, werden bei „Information“ alle verfügbaren Informationen geloggt.

### Wichtiger Hinweis

**Das Schreiben von Tracing-Informationen ist teuer. Setzen Sie das LogLevel mit Bedacht! Im normalen Betrieb sollte unbedingt die Einstellung: „Error“ gewählt sein.**

### File Logger

- schreibt die Tracing-Informationen in eine Datei. Der File Logger ist per Default in allen Konfigurationsdateien eingetragen.

### Console Logger

- schreibt die Tracing-Informationen in die Konsole. Der Console Logger ist per Default in „Sagede.ApplicationServer.ConsoleHost.exe.config“ und „Sagede.ApplicationServer.IsolationProcess.exe.config“ eingetragen.

Das Tracing via System Diagnostics, wie oben beschrieben, wird per Default nicht mehr verwendet, kann aber über die Konfiguration wieder eingeschaltet werden.

## Tracing via Ereignisablaufverfolgung für Windows (ETW)

Ereignisablaufverfolgung für Windows (Event Tracing for Windows – ETW) ist eine allgemein gültige, schnelle Ablaufverfolgung, die vom Betriebssystem bereitgestellt wird. Mit einem im Kernel implementierten Puffer- und Protokollmechanismus stellt ETW einen Ablaufverfolgungsmechanismus für Ereignisse bereit, die von Anwendungen im Benutzermodus oder von Gerätetreibern im Kernelmodus ausgelöst werden. Zusätzlich gibt ETW die Möglichkeit, die Protokollierung dynamisch zu aktivieren und zu deaktivieren, sodass detaillierte Ablaufverfolgungen in Produktionsumgebungen einfach ohne Neustarts oder Anwendungsneustarts durchgeführt werden können.

Als Anzeige- und Administrationswerkzeug für das ETW-Tracing, als auch für das Komponenten-Tracing, wird der Sage TraceLogManager verwendet. Dieser kann über das Startmenü gestartet werden.

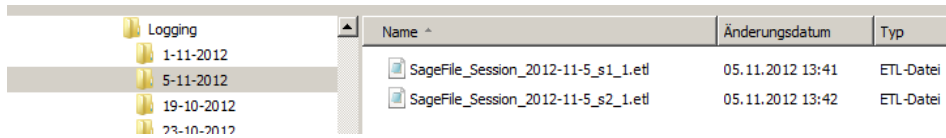
Das ETW-Tracing des Applikationsservers schreibt die Information zum einen in eine Logging-Datei. Die Logging-Datei ist eine binäre Datei und kann nicht von einem normalen Textprogramm geöffnet werden. Das Format ist ein Standardformat für vom Windows-System erzeugte Logdateien, die Dateiendung ist „.etl“. Windows stellt verschiedene Tools und APIs bereit um diese Dateien zu lesen und in xml, csv oder andere Formate zu konvertieren. Der TraceLogManager enthält eine Implementierung die damit ebenfalls umgehen kann, sodaß ihr Inhalt im TraceLogManager auf normale Weise angezeigt werden kann.

Neben dem Datei-Logging können die Logging-Informationen auch in Echtzeit via TraceLogManager angezeigt werden.

Bitte beachten Sie, daß die Computerverwaltung nicht zum Editieren und zum Visualisieren der Einstellungen des ETW-Tracing des Applikationsservers verwendet werden kann,

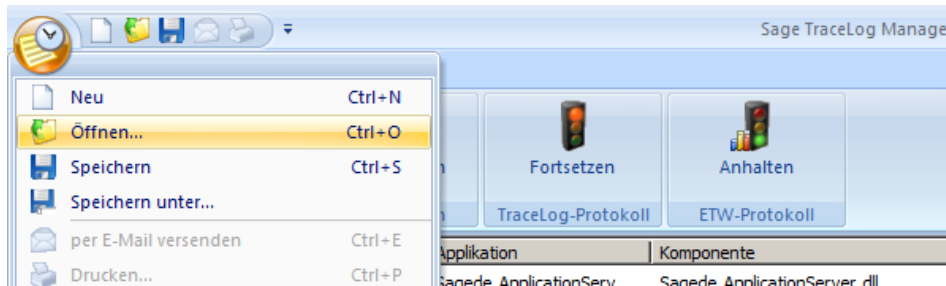
## Datei-Logging via ETW

Die Dateiausgabe erfolgt in einen Ordner, der das Tagesdatum als Namen hat, an dem das Logging gestartet wurde. Wird es an einem Tag mehrmals gestartet, so befinden sich alle Dateien im gleichen Ordner. Existiert noch kein Ordner für den betreffenden Tag, wird dieser automatisch angelegt. Diese Ordner wiederum befinden sich im Logging-Ordner des Applikationsservers.



Name	Änderungsdatum	Typ
SageFile_Session_2012-11-5_s1_1.etl	05.11.2012 13:41	ETL-Datei
SageFile_Session_2012-11-5_s2_1.etl	05.11.2012 13:42	ETL-Datei

Die Protokoll-Dateien beginnen mit „SageFile\_Session“ und bekommen das Datum, sowie einen Zähler angehängt. Dieser ist nötig, falls mehrere Dateien an einem Tag erzeugt werden. Die Log-Dateien besitzen die Endung .etl. Zur Analyse dieser Log-Dateien, können sie im TraceLogManager geöffnet werden.



## Konfiguration des ETW Datei-Loggings

Über die Konfigurationsdatei des Applikationsservers kann das permanente Logging gesteuert werden. Dafür ist der folgende Eintrag in Datei Sagede.ApplicationServer.Core.config nötig:

```
<?xml version="1.0" encoding="utf-8" ?>
<server>
..
<diagnostics>
<trace>
<providers>
<provider name="Sagede Application Server
    guid="08F36BC6-D38D-45BC-99DB-C995729A5D0E"
    level="Warning"
    keywords=""
/>
</providers>
</trace>
</diagnostics>
</server>
```

### @name

Benutzerfreundlicher Name des Providers, der von Diagnosetools angezeigt werden kann

### @guid

ID des Providers, mit der dieser auf dem Rechner registriert ist. Beim Starten des Applikationsservers registriert sich der Provider, beim Beenden, wird er wieder deregistriert. Der Wert für den Provider des Applikationsservers ist "08F36BC6-D38D-45BC-99DB-C995729A5D0E".

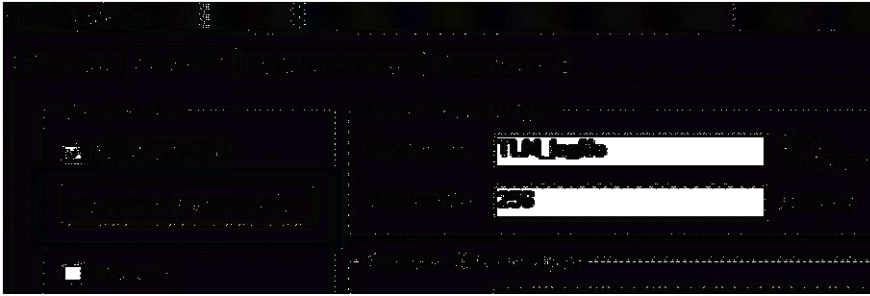
### @level



Bestimmt den Grad an Information, der protokolliert wird. Mögliche Werte sind: Critical, Error, Warning, Information, Verbose. Der Default-Wert ist „Warning“.

Nach einer Änderung in der Konfigurationsdatei ist ein Neustart des Servers erforderlich, damit sie wirksam wird.

Die Parameter für das erweiterte Datei-Logging können im TraceLogManager geändert werden. Das geht allerdings nur, wenn das Logging bereits gestartet ist, erst dann ist die Schaltfläche „Erweiterte Dateiausgabe“ aktiviert.

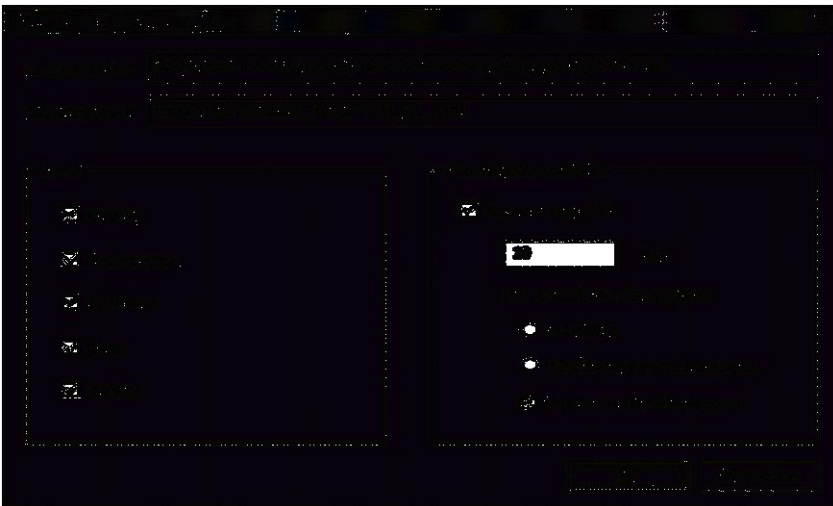


Klickt man darauf, wird ein neuer Dialog angezeigt, der Informationen über das aktuell aktive Dateiloggung enthält und die Möglichkeit bietet, verschiedene Parameter zur Laufzeit des Logging zu ändern.

Angezeigt wird

- der aktuelle Ausgabepfad
- der Name der aktuellen Logdatei
- das eingestellte Logging Level
- ob die Begrenzung für die maximale Dateigröße aktiv ist
- der Wert für die maximale Dateigröße
- das Verhalten, wenn die maximale Dateigröße erreicht ist.

Die Punkte 3 bis 6 können konfiguriert werden. Wirksam werden die Änderungen, sobald dieser Dialog und der Einstellungen-Dialog mit OK abgeschlossen werden. Die Änderungen gelten bis der Applikationsserver beendet wird. Bei einem Neustart des Applikationsservers gelten wieder die Einstellungen des Levels aus der Config-Datei „Sagede.ApplicationServer.Core.config“ des Applikationsservers.



Die Einstellungen haben folgende Bedeutung:

#### Level

Die Level können nicht einzeln ein- und ausgeschaltet werden, sondern umfassen immer alle schwerwiegenderen Meldungen. Wird also Warning eingeschaltet, dann ist automatisch auch Error und Critical aktiviert. Wird Debug eingeschaltet, dann sind alle anderen Level ebenfalls aktiviert.

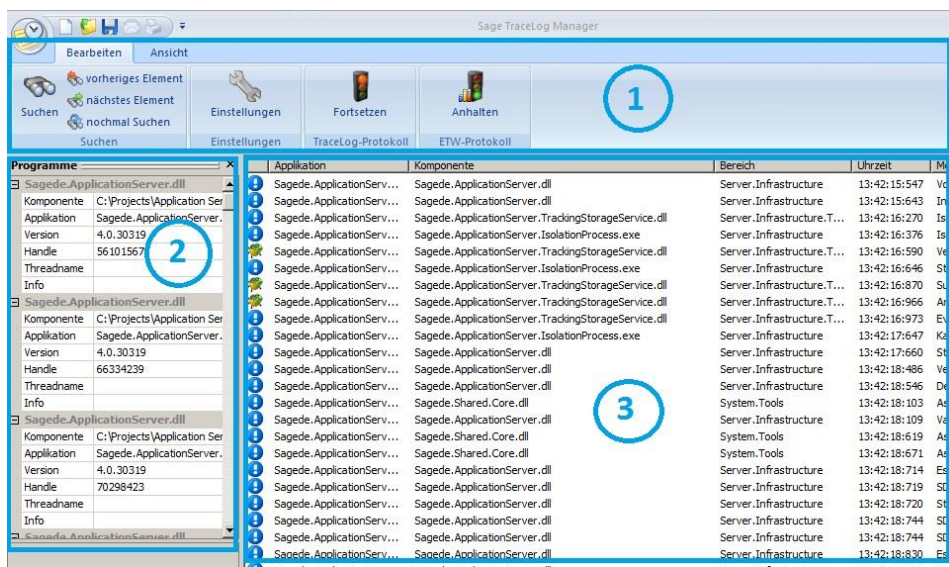
## Maximale Dateigröße

Wenn die Begrenzung nicht aktiviert ist, bildet der Speicherplatz der Festplatte die einzige Beschränkung. Definiert man eine Begrenzung und ist diese erreicht, dann kann man das weitere Verhalten ebenfalls steuern. Folgende Möglichkeiten stehen zur Wahl:

- Anhalten  
Das Logging stoppt bei Erreichen der gewählten Dateigröße
- Wieder von vorne beginnen  
Das Protokoll wird am Dateianfang durch Überschreiben alter Einträge fortgesetzt
- In neuer Datei fortsetzen  
Es wird jeweils bei Erreichen der Grenze eine neue Datei angelegt, der Dateiname schließt mit einem Index, der mit jeder neuen Datei erhöht wird.

## Echtzeit-Logging via ETW und TraceLogManager

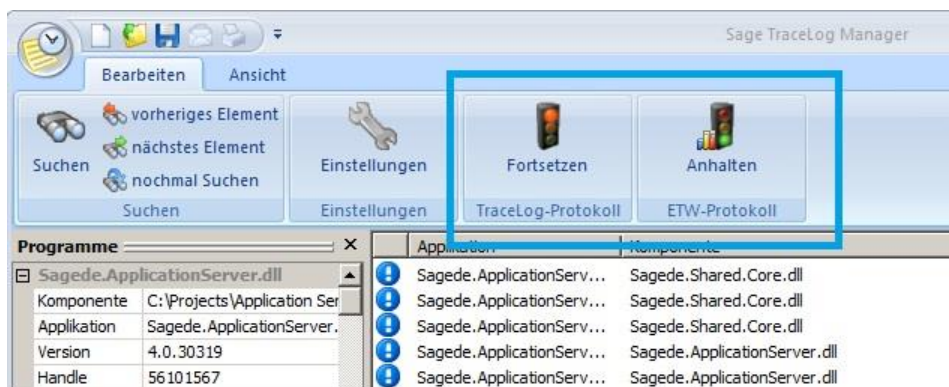
Zur Aktivierung des Echtzeit-Loggings ist lediglich der Start des TraceLogManagers erforderlich. Es werden dann sofort alle Logging-Events empfangen und angezeigt. Neben ETW-Ereignissen werden auch Ereignisse des TraceLog-Loggings entgegengenommen. Der TraceLogManager muss als Administrator gestartet werden.



Der TraceLog Manager besteht prinzipiell aus drei Bereichen:

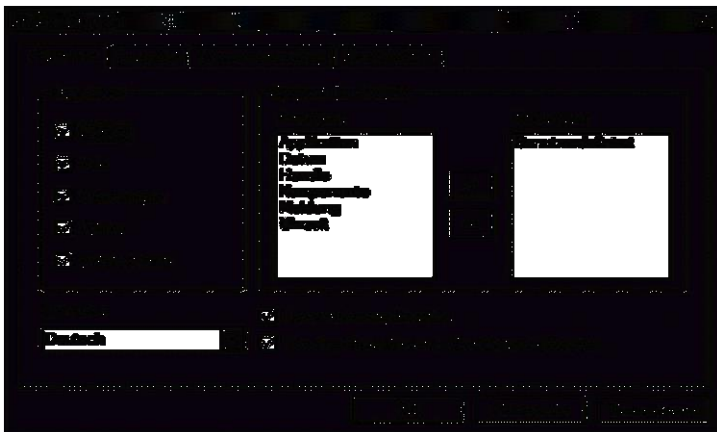
- Über den Ribbon können Einstellungen geändert werden und das Logging gestartet und angehalten werden.
- Im linken Bereich werden die Programme bzw. die Komponenten angezeigt, die Meldungen erzeugt haben.
- Im rechten Bereich werden die Meldungen angezeigt.

Das Echtzeit-Logging aber auch das TraceLog-Logging kann über die Buttons im Ribbon angehalten und gestartet werden.



In einer produktiven Server Umgebung sollte nur das ETW-Logging aktiv sein, da es bei großem Log-Volumen in Kombination mit gestartetem TraceLog-Logging zu Deadlock Situationen kommen kann. Per Default ist beim Start des TraceLogManager das TraceLog-Logging angehalten.

Das Logging Level für die Echtzeitsicht kann zur Laufzeit des Servers geändert werden. Dazu ruft man den Einstellungen-Dialog auf, um den gewünschten Wert zu setzen.



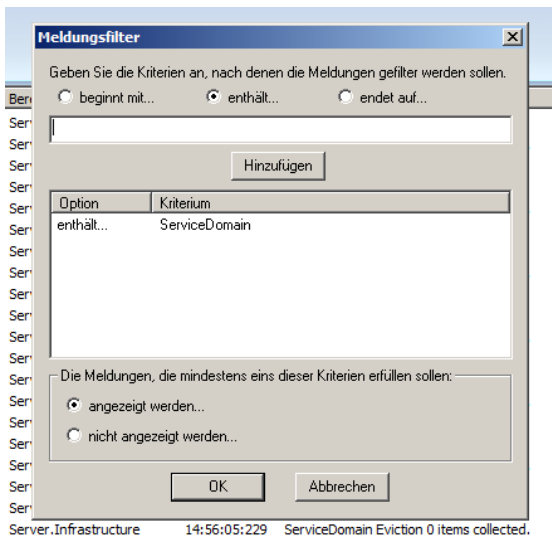
Zusätzlich kann über ein Kontextmenü auch die Echtzeitsicht bezüglich der angezeigten Loggingeinträge konfiguriert werden. Man kann nach dem Level im Meldungsfilter filtern, und sich so z. B. nur die Warnings anzeigen lassen. Es wird aber intern weiterhin gemäß dem Einstellungen-Dialog protokolliert.

Applikation	Komponente	Bereich	Uhrzeit	Meldung	
Sagede.Ap...	Sagede.ApplicationServer.IsolationProcess.exe	Server.Infrastructure	14:09:19:747	Starte Serv	
Sagede.Ap...	Sagede.ApplicationServer.TrackingStorageService.dll	Server.Infrastructure.Tracking	14:09:20:355	Suche nad	
Sagede.Ap...	Sagede.ApplicationServer.TrackingStorageService.dll	Server.Infrastructure.Tracking	14:09:20:147	Anzahl Tab	
Sagede.Ap...	Sagede.Appl	Service.dll	Server.Infrastructure.Tracking	14:09:20:153	Eviction für
Sagede.Ap...	Sagede.Appl	structure	14:09:20:517	Kanal in de	
Sagede.Ap...	Sagede.Appl	structure	14:09:20:524	Stelle Verbi	
Sagede.Ap...	Sagede.Appl	structure	14:09:20:904	Verbindung	
Sagede.Ap...	Sagede.Shar	s	14:09:21:892	Assembly r	
Sagede.Ap...	Sagede.Shar	s	14:09:21:204	Assembly r	
Sagede.Ap...	Sagede.Shar	s	14:09:21:230	Assembly r	
Sagede.Ap...	Sagede.ApplicationServer.dll	structure	14:09:21:363	Der Trackir	
Sagede.Ap...	Sagede.ApplicationServer.dll	structure	14:09:21:381	Es wurde d	
Sagede.Ap...	Sagede.ApplicationServer.dll	structure	14:09:21:393	Es wurde d	
Sagede.Ap...	Sagede.ApplicationServer.dll	structure	14:09:21:410	Validation s	
Sagede.Ap...	Sagede.ApplicationServer40.OfficeLine.D	river	14:09:21:699	ServiceCor	
Sagede.Ap...	Sagede.ApplicationServer.dll	Server.Infrastructure	14:09:21:750	Es wurde d	
Sagede.Ap...	Sagede.ApplicationServer.dll	Server.Infrastructure	14:09:21:752	SData wird	

Zusätzlich lassen sich die Einträge im Ereignis-Bereich nach Komponente und Bereich filtern. Dies ist insbesondere bei der Analyse von vielen Ereignissen hilfreich. Ein Filter über den Inhalt des Feldes „Meldung“ steht ebenfalls zur Verfügung.

Applikation	Komponente	Bereich	Uhrzeit	Meldung	
Sagede.Ap...	Sagede.ApplicationServer.Is...	Server.Infrastructure	14:09:16:436	IsolationProcess '04bceec5-ee4f	
Sagede.Ap...	Sagede.ApplicationServer.Is...	Server.Infrastructure	14:09:16:636	Starte ServiceHost für den Isola	
Sagede.Ap...	Sagede.ApplicationServer.Is...	Server.Infrastructure	14:09:17:416	Kanal in den IsolationProcess '04	
Sagede.Ap...	Sagede.Appl	Server.dll	Server.Infrastructure	14:09:17:420	Stelle Verbindung mit IsolationPr
Sagede.Ap...	Sagede.Appl	rastructure	14:09:17:713	Verbindung mit IsolationProcess	
Sagede.Ap...	Sagede.Appl	ols	14:09:17:818	Assembly resolved: 'c:\sage\ol\%	
Sagede.Ap...	Sagede.Appl	ols	14:09:18:598	Assembly resolved: 'c:\sage\ol\%	
Sagede.Ap...	Sagede.Appl	ols	14:09:18:180	Assembly resolved: 'c:\sage\ol\%	
Sagede.Ap...	Sagede.Appl	rastructure	14:09:18:327	Es wurde der Service Sagede.Ap	
Sagede.Ap...	Sagede.Appl	rastructure	14:09:18:344	Es wurde der Service Sagede.Ap	
Sagede.Ap...	Sagede.Applicatio	Driver	14:09:18:470	ServiceContext erzeugt.	
Sagede.Ap...	Sagede.Applicatio	rastructure	14:09:18:881	Es wurde der Service Sagede.Ap	
Sagede.Ap...	Sagede.Applicatio	rastructure	14:09:18:883	Isolationsprozess für ArtikelServ	
Sagede.Ap...	Sagede.Applicatio	rastructure	14:09:18:884	Starte IsolationProcess...	
Sagede.Ap...	Sagede.Applicatio	rastructure	14:09:18:885	Vorinitialisierung der konfiguriert	
Sagede.Ap...	Sagede.ApplicationServer.dll	Server.Infrastructure	14:09:18:952	Initialisiere Systemdienste...	

In dem folgenden Dialog kann der Inhaltsfilter erstellt, geändert und gelöscht werden.



## Performance-Counter

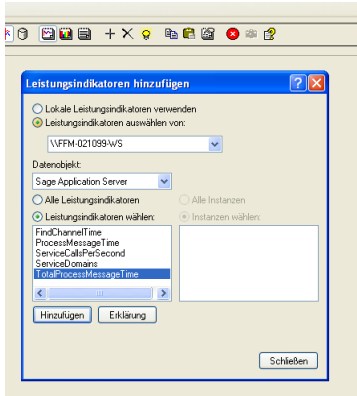
Performance-Counter müssen installiert werden. Dies erfolgt über das Applikationsserver-Setup. Sie liefern Daten für den standardmäßig auf allen Rechnern installierten Performance Monitor. Die Perfmon.exe kann über Run/Ausführen aufgerufen werden.

Es existieren folgende Counter:

- **ActiveServiceDomains**  
Liefert die Anzahl der aktuell aktiven ServiceDomains ~ Anzahl der aktuellen parallelen Requests.
- **AsyncQueuedRequests**  
Anzahl der asynchronen Anfragen, die sich in der Warteschlange des Applikationsservers befinden.
- **AsynchronWorkingRequests**  
Anzahl der asynchronen Anfragen, die aktuell abgearbeitet werden.
- **DeniedServiceCalls**  
Anzahl der aufgrund fehlgeschlagener Authentifizierung zurückgewiesenen Anfragen.
- **DeniedServiceCallsPerSecond**  
Anzahl der aufgrund fehlgeschlagener Authentifizierung zurückgewiesenen Anfragen pro Sekunde.
- **FailedServiceCalls**  
Liefert die Anzahl der Anzahl der zurückgewiesenen Anfragen aufgrund eines internen Fehlers.
- **FailedServiceCallsPerSecond**  
Liefert die Anzahl der Anzahl der zurückgewiesenen Anfragen aufgrund eines internen Fehlers per Sekunde.
- **FindChannelTime**  
Durchschnittszeit, die die Infrastruktur braucht um eine freie Service-Domain zu finden.
- **ProcessMessageTime**  
Durchschnittszeit, die die gehostete Applikation für den Aufruf benötigt.
- **RejectedServiceCalls**  
Liefert die Anzahl der zurückgewiesenden Requests aufgrund hoher Last.
- **RejectedServiceCallsPerSecond**  
Liefert die Anzahl der zurückgewiesenden Requests aufgrund hoher Last per Sekunde.
- **ServiceCallsPerSecond**  
Durchschnittliche Anzahl von Serviceaufrufen per Sekunde.
- **ServiceDomainCreationRejected**  
Liefert die Anzahl von fehlgeschlagenen Versuchen eine neue ServiceDomain zu erzeugen.
- **ServiceDomainPoolSize**  
Liefert die Anzahl der ServiceDomains, die in dem ServiceDomain-Pool liegen.

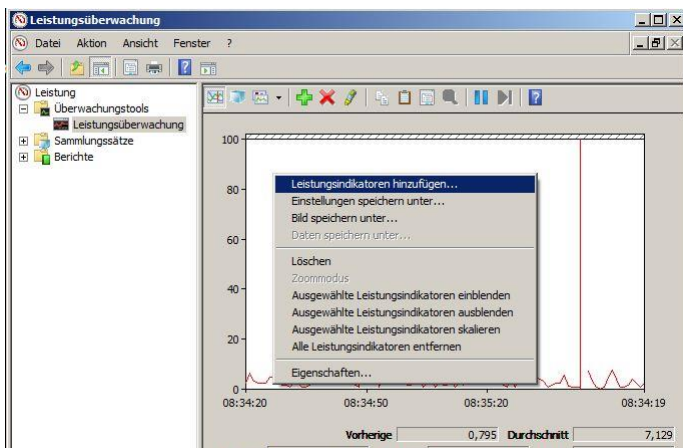
- ServiceDomains  
Gesamtzahl der aktuell im Server vorhandenen ServiceDomains (inklusive der ServiceDomains, die im Pool liegen).
- TotalProcessMessageTime  
Durchschnittliche Gesamtdauer eines Serviceaufrufs.

Die Performance-Counter müssen, wie alle anderen Performance-Counter eines Systems, im Performance Monitor extra aktiviert werden. Dies geschieht durch Anklicken des „+“ –Zeichens oben in der Toolbar → Auswählen von „Sage Applikationsserver“ als Datenobjekt → Hinzufügen der gewünschten Counter aus der Liste.

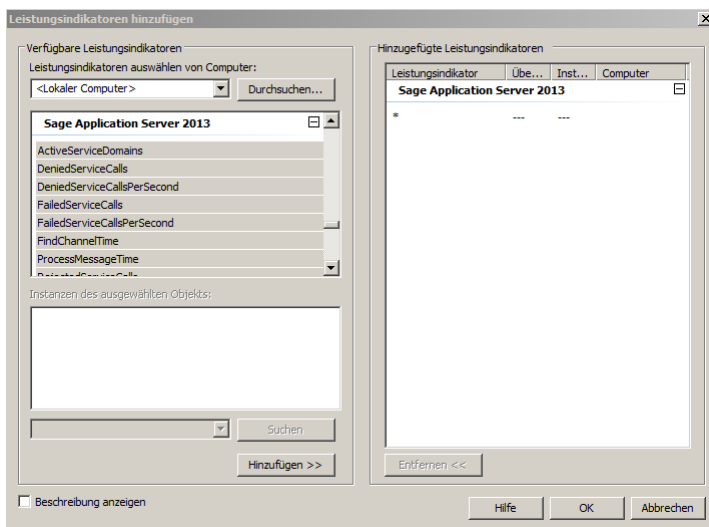


## Unter Windows 7

Nach dem Start der Leistungsüberwachung: Rechte Maustaste -> Leistungsindikatoren hinzufügen



Gewünschte Leistungsindikatoren des Applikationsservers finden sich unter „Sage Application Server 2013“



Leistungsindikatoren markieren und „Hinzufügen“ klicken.

## Server Manager

Der mit dem Applikationsserver installierte Server Manager stellt die administrative Oberfläche für Verwaltung der Infrastruktur zur Verfügung. Diese sind:

- Initialisiert / deinitialisiert Einrichtung der Konnektivität via Communication Service
- Startet / stoppt Connectivity Client via Communication Service
- Konfiguriert Connectivity Client via Communication Service

### Plugin: "Lokale Maschine"

- Connectivity Dienste in „Dienste“ anzeigen.
- Status des Connectivity Client Dienstes kann hier nicht geändert werden.

### Plugin: "Installation"

- Neues Plugin "Installation" für den Server Manager, über das Installations-weite administrative Aufgaben erledigt werden können.
- Neuer Menüpunkt: "API"
  - In Tabellenform werden alle in der RemoteRegistry registrierten Communication Services mit ihren Stati dargestellt.

Server	Status	Verbindung	Verbunden mit
<a href="#">Ffm-dfsfsfsf</a>	 Erreichbar	aktiv   	<a href="#">Ffm-trwtetret</a>
<a href="#">Ffm-khhjhkhk</a>	 Erreichbar	inaktiv  	
<a href="#">Ffm-gdgerte</a>	Gestoppt		

- Über das "Edit"-Icon kann man die Konfiguration des jeweiligen Communication Services bearbeiten.



Sage

©2024 The Sage Group plc or its licensors. All rights reserved. Sage, Sage logos, and Sage product and service names mentioned herein are the trademarks of Sage Global Services Limited or its licensors. All other trademarks are the property of their respective owners.