[accantum] Application Programming Interface
Kurzanleitung

Accantum GmbH
Äußere Oberaustraße 36/4
Anbau-Nord 1.OG
D-83026 Rosenheim

Titel:	[accantum] Application Programming Interface
Betreff:	Kurzanleitung
Erstelldatum:	09.08.2018; HLE
Letzte Bearbeitung:	13.08.2018 12:52:00, FWI
Revisionsnummer:	1.0
Speicherort:	API Kurzanleitung.docx

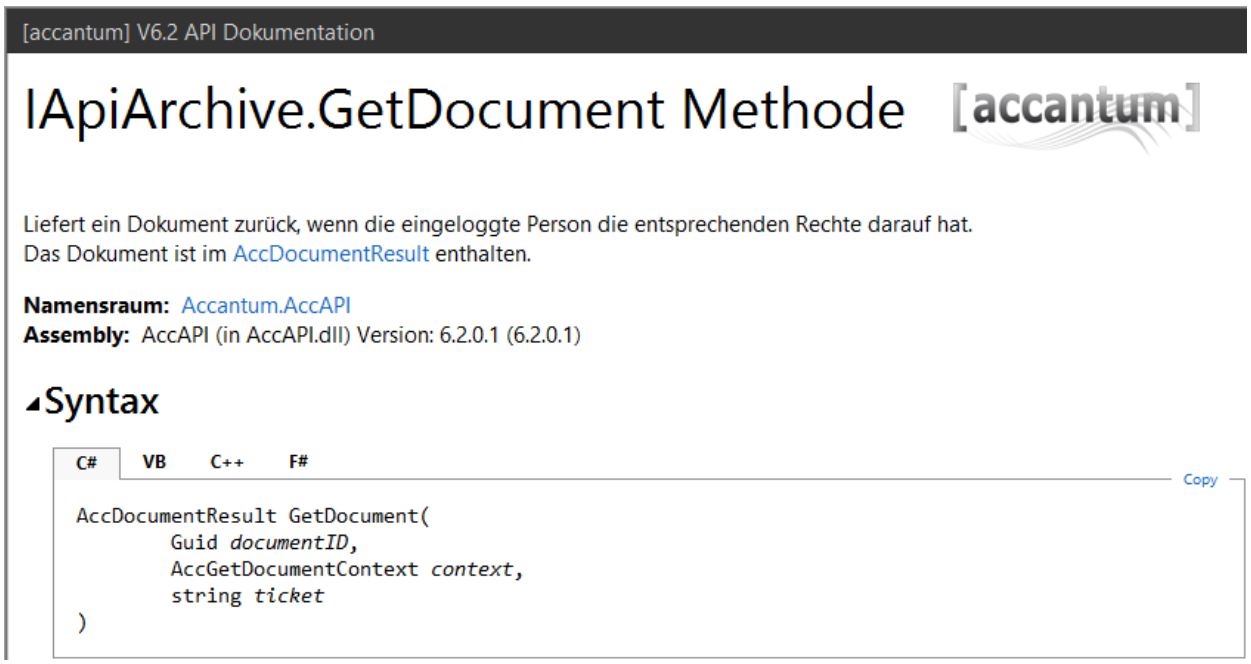
Inhalt

1 Einleitung

Dieses Dokument ist für Software Entwickler verfasst, die sich einen ersten Überblick über die API von [accantum] und deren Möglichkeiten verschaffen möchten.

Das [accantum] Archivsystem stellt Ihnen über einen WebService eine Standard API zur Verfügung, über die Sie Archivfunktionalitäten in Ihr bestehendes System integrieren können.

Über diese API erhalten Sie Zugriff auf die nahezu gesamte Funktionalität des [accantum] Archivsystems.


[accantum] V6.2 API Dokumentation

IApiArchive.GetDocument Methode [accantum]

Liefert ein Dokument zurück, wenn die eingeloggte Person die entsprechenden Rechte darauf hat.
Das Dokument ist im AccDocumentResult enthalten.

Namensraum: Accantum.AccAPI
Assembly: AccAPI (in AccAPI.dll) Version: 6.2.0.1 (6.2.0.1)

Syntax

C# VB C++ F# Copy

```
AccDocumentResult GetDocument(
        Guid documentID,
        AccGetDocumentContext context,
        string ticket
)
```

Die [accantum] API unterstützt neben C#, VB, C++ und F#. Es können weitere Programmiersprachen verwendet werden, mit denen SOAP-Nachrichten versendet und empfangen werden können.

SOAP-Nachrichten sind Nachrichten im XML-Format und werden über das http-Protokoll versendet.

1.1 Ressourcen

Folgende Dokumentation / Ressourcen stellen wir für die Entwicklung zur Verfügung.

- [accantum] V6.3 API Dokumentation.chm (Helpfile)
- Beispiel-Projekt „WCFClientSample“

Beispiel

Codebeispiel zur Erstellung eines Tickets:

C#

Copy

```
private string CreateTicket(string a_strUser, string a_strPwd, string a_strSystemIdentifier)
{
      string strCredentials = string.Format("{0}:{1}", a_strUser, a_strPwd);
      byte[] oBytes = Encoding.UTF8.GetBytes(strCredentials);
      string strEncodedCredentials = Convert.ToBase64String(oBytes);

      string strTemp = strEncodedCredentials.Replace('+', '-').Replace('/', '_').TrimEnd('=');
      string strTicket = string.Format("{0}@{1}", strTemp, a_strSystemIdentifier);
      return strTicket;
}
```

Hinweise:

Das Helpfile *.chm ist im Download von [accantum] im Ordner Dokumentation enthalten.

Das Helpfile *.chm muss auf einem lokalen Laufwerk installiert werden.

2 Entwicklungsumgebung

Für die Nutzung der API ist eine Installation der [accantum] Dienste notwendig. Mit der mitgelieferten Demo-Lizenz ist ein Demo-Konnektor freigeschaltet, mit welchem die API verwendet werden kann. Für die Verwendung mit einer „scharfen" Lizenz muss ein eigenes Modul bei Accantum beantragt werden.

Die API besteht aus der Web- und WCF-API.

[accantum] V6.3 kann über die Homepage www.accantum.de runtergeladen werden (Der Bereich Downloads steht nur registrierte Benutzern zur Verfügung)

Die Web-API besteht lediglich aus einer Web-Url, mit der je nach angegebenen Parametern ein Dokument gesucht oder geöffnet werden kann. Für die Nutzung der Web-API ist eine vollständig installierte Accantum-Version notwendig (Web-GUI und Dienste). Im Gegensatz zur WCF-API sind hier keine Programmierkenntnisse notwendig. Um z.B. ein Dokument anzuzeigen, ist lediglich der Aufruf einer URL nötig. Als Parameter werden die Authentifizierungsart, die WCF-Service-URL und die Dokument-ID angegeben:

```
http://server/accantum/webapi?auth=windows&archive=http%3a%2f%2fserver%3a50000%2f&docid= FABF5D84-5BE4-47d8-BCEB-274CDEBBF67D
```

Detailliertere Informationen erhalten Sie in unserer API-Dokumentation ([accantum] V6.0 API Dokumentation.chm) unter „How To's > Verwendung der Web-API".

Die WCF-API wird durch unseren Windowsdienst „AccServices.exe" gehostet. Mit dem [accantum] Configuration-Manager können Sie den Port einstellen, unter dem der Dienst zur Verfügung steht. Bei einer Standard-Installation mit nur einem Archiv wird die API unter der Adresse http://server:50000 gehostet. Die WCF-API bildet die vollständige Accantum-Funktionalität ab.

Mit einer .NET Programmiersprache kann im Microsoft Visual Studio relativ einfach ein WCF-Service in ein eigenes Projekt eingebunden werden. Durch das Hinzufügen des WCF-Service werden automatisch Proxy-Klassen generiert, über die dann die Schnittstelle benutzt werden kann. Sollte Ihre Programmiersprache keine Proxy-Klassen aus einem WCF-Service erzeugen können, lässt sich unsere API mit etwas mehr Aufwand auch direkt per SOAP nutzen.

3 WCF-API: Erste Schritte

Wurde die Service-Reference in das Projekt eingebunden, dann kann in wenigen Schritten die API genutzt werden. Die meist benötigten Aufrufe sollen hier kurz aufgeführt werden. Code-Beispiele hierzu finden Sie in unserer kleinen Anwendung „WCFClientSample", in der noch weitere Funktionalitäten rudimentär implementiert sind.

- Einbinden der Service-Referenz
- Rückgabewert von API-Methoden
- Authentifizierung
- Stammdaten abrufen
- Dokument-Metadaten aufbereiten und speichern
- Binärdaten übertragen
- Metadaten eines Dokuments abrufen
- Binärdaten eines Dokuments abrufen
- Suche nach einem Dokument
- Dokument ablegen
- Accantum-Queue auslesen

3.1 Einbinden der Service-Referenz (.NET)

Bevor die API in eigenen (.NET-)Anwendungen genutzt werden kann, muss zum Projekt eine Service-Referenz hinzugefügt werden. Dadurch werden automatisch Proxy-Klassen erstellt, so dass über ein Objekt-Modell auf die Methoden und Klassen der API zugegriffen werden kann.

In anderen Programmiersprachen sollte dies ähnlich funktionieren, bei Delphi ist uns bekannt, dass hier bei den automatisch erstellten Proxy-Klassen manuell nachgebessert werden muss.

Bei Programmiersprachen, welche keine automatische Erstellung von Proxy-Klassen unterstützen, haben Sie immer noch die Möglichkeit, die erforderlichen SOAP-Nachrichten manuell zu erstellen.

3.2 Rückgabewert von API-Methoden

Alle API-Methoden geben eine Klasse vom Typ *AccResult* oder eine davon abgeleitete Klasse zurück. Über die Property *HasErrors* bzw. *HasWarnings* kann abgefragt werden, ob ein Fehler aufgetreten ist. Im Falle eines Fehlers bzw. einer Warnung können eine oder mehrere Fehlermeldungen über das Property *MsgCollection* ausgelesen werden (Array der Klasse *AccMsgItem*).

3.3 Authentifizierung

Bei jedem Aufruf einer API-Methode muss ein Ticket übergeben werden. Dieses Ticket setzt sich aus einer Session-ID und einem Modul-Identifier zusammen. Die Session-ID erhält man durch die Authentifizierung mit der Methode *Login()*. Der Modul-Identifier wird fest von uns vorgegeben und lautet für den **Demo-Konnektor**: „*4uaQ7b0lwdqT7xyIHU15jg*". Der Demo-Konnektor ist nur in einer Demo-Lizenz freigeschaltet. Für eigene Module muss der Identifier und damit auch das Modul bei Accantum beantragt werden (siehe auch Kap. **Fehler! Verweisquelle konnte nicht gefunden werden.**).

Ein Beispiel für das Erstellen eines Tickets finden Sie auch im Helpfile unter „How To's > Authentifizierung".

Die Dauer der Gültigkeit eines Tickets kann in der [accantum] Management Console konfiguriert werden (Default: 20 Minuten?)

3.4 Stammdaten abrufen

Wird ein Dokument archiviert, dann muss z.B. auch eine Kategorie gesetzt werden, welche in den Stammdaten gepflegt wird. Mit *GetLists(ObjectTypes)* können Stammdaten abgerufen werden, wenn nur einfache Informationen wie ID und Name (z.B. für die Anzeige in einer ComboBox) benötigt werden. Über Parameter wird angegeben, welche Art von Stammdaten ausgelesen werden soll.

Für das Abrufen von detaillierteren Informationen gibt es für viele Stammdaten eigene Get-Methoden. Zum Beispiel ist die Methode *GetAttributeDefs()* zum Abrufen der Attribut-Definitionen besser geeignet, da hier auch die List-Items von Auswahllisten enthalten sind.

3.5 Dokument-Metadaten speichern

Ein Dokument in Accantum besteht aus einem Metadaten-Satz für das Dokument (*AccDocument*), einem Content (*AccContent*) und einem Ablagepfad (*AccPath*). Um ein Dokument zu archivieren müssen alle Objekte erstellt werden. Erst wenn diese Objekte mit der Methode *StoreDocument(Doc)* gespeichert wurden, können die Binärdaten übertragen werden.

Ein Beispiel finden Sie auch im Helpfile unter „How To's > Archivierung eines Dokuments".

3.6 Dokument-Attribute ändern

Wird ein Dokument abgerufen, enthält es nur diejenigen benutzerdefinierte Attribute, für welche Werte existieren. Deshalb bietet sich an, alle für das Dokument möglichen Attribute über die Methode *GetAttributeDefs(Param)* abzurufen. Im Parameter der Methode muss die Kategorie des Dokuments angegeben werden. Sollen nur Dokumenttyp spezifische Attribute zurückgegeben werden, dann kann auch zusätzlich der *DocTyp* angegeben werden (z.B. Attribut für Empfänger bei E-Mails).

Soll ein neuer Wert der Attribut-Auflistung des Dokuments hinzugefügt werden, muss je nach Datentyp eine Ableitung vom Typ *AccBaseValue* (z.B. *AccStringValue, AccIntValue*, etc.) der Attribut-Auflistung hinzugefügt werden.

3.7 Binärdaten übertragen

Da die WCF-Requests einer Größenbeschränkung unterliegen, können die Binärdaten in einer Schleife „Häppchenweise" übertragen werden. Mit der Übertragung des letzten Häppchens muss die Übertragung als *commited* gekennzeichnet werden.

Ein Beispiel finden Sie auch im Helpfile unter „How To's > Archivierung eines Dokuments".

Weitere Beispiele finden Sie in unserer kleinen Beispiel-Anwendung *WCFClientSample*, in der ein paar wesentliche Funktionalitäten rudimentär implementiert sind.

3.8 Metadaten eines Dokuments abrufen

Soll auf ein bereits archiviertes Dokument zugegriffen werden, können mit der Methode *GetDocument(Id, Context)* die Metadaten eines Dokuments abgerufen werden. Beim Parameter *Context* kann angegeben werden, ob nur bestimmte Daten des Dokuments zurückgeliefert werden sollen (wenn null, dann werden alle Daten zurückgegeben).

Einige der Attribute sind direkt über das Objekt *AccDocument* zugreifbar (z.B. Dokumentnummer, Name, Beschreibung, etc.). Benutzerdefinierte Attribute sind in der Auflistung *Attributes* in Form eines Arrays vom Typ *AccBaseValue[]* enthalten.

3.9 Binärdaten eines Dokuments abrufen

Zum Abrufen des Original-Dokuments müssen zunächst die Contents des Dokuments mit der Methode *GetContents(DocId, Version)* abgerufen werden. Nun können mit der Methode *GetBinaryData(DocId, Content)* die Binärdaten des Original-Dokuments abgerufen werden, in dem der erste Content mit *ContentTyp = Original* als Parameter übergeben wird.

Zu beachten ist bei der Methode *GetBinaryData()*, dass die Binärdaten in einer Schleife abgerufen werden müssen, bis alle Binär-Schnipsel heruntergeladen sind.

3.10 Suche nach einem Dokument

Natürlich kann auch nach Dokumenten gesucht werden. Mit der Methode „FindDocuments()“ wird zunächst einmal die Suche angestoßen, und liefert ein erstes Ergebnis zurück. Da eine Suche viele Ergebnisse zurückliefern kann, welche die technischen Grenzen der Übertragungsmenge sprengen können, verwenden wir einen Paging-Mechanismus, um immer nur einen Teil des Suchergebnisses zurückzugeben (Umfang kann bei den Parametern der Parameter angegeben werden). Mit der Methode „FindMordDocuments()“ lassen sich dann weitere Suchergebnisse abrufen. Der hier notwendige Parameter „SearchId“ wird beim Aufruf der Methode „FindDocuments()“ zurückgegeben. Ob eine „erweiterte Suche“ oder eine Attribut-Suche ausgeführt werden soll, wird über die Suchparameter gesteuert.

3.11 Dokument ablegen

Soll ein Dokument in einen endgültigen Ablageort verschoben oder das Kennzeichen „revisionssicher“ gesetzt werden, so wird dies mit der Methode „BulkEdit()“ durchgeführt.

3.12 Accantum-Queue auslesen

Mit der Methode *GetNextQueueEntries(Count)* können benachrichtigte Dokumente aus der Accantum-Queue gelesen werden. Diese Methode liest nur unverarbeitete Einträge aus. Mit dem Aufruf der Methode wird der Status automatisch auf *Processing* gesetzt, so dass dieser Eintrag mit dem nächsten Aufruf nicht mehr ausgelesen wird.

Nach erfolgreicher Verarbeitung des Dokuments muss der Queue-Eintrag auf den Status *Finished* oder *Failure* gesetzt werden. Hierzu wird das Property *State* des ausgelesenen Queue-Eintrags auf den entsprechenden Status gesetzt.

Mit der Methode *StoreQueueEntries(QueueEntry[])* kann der geänderte Status gespeichert werden.

4 Programmierbeispiel - Sample

Auf Anfrage stellen wir Ihnen gerne ein Programmierbeispiel zur Verfügung.

Bitte wenden Sie sich dazu an: info@accantum.de

Voraussetzung:

- Lauffähiges Accantum (Demo-Version)
- Accantum-Lizenz mit freigeschaltetem Modul „Demo-Konnektor“

Technologie:

- .NET Framework
- Programmiersprache C#
- Anbindung an WCF-Schnittstelle der Accantum-Dienste

Das Beispiel-Projekt "WCFClientSample" enthält folgende Funktionalität:

- Login/Logout
- Abrufen von
  - Ablageorte
  - Kategorien
  - Ablageregeln
- Archivieren eines Dokuments
- Suche nach Dokumenten
- Abrufen der Dokument-Metadaten (inkl. Benutzerdefinierte Attribute)
- Abrufen des Dokument-Binärdaten (zur Anzeige)
- Neue Version erstellen