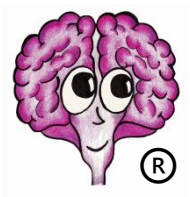
PTI-DB-00023
Revision 10

DocuBrain® TechDoc SharePoint Business Connectivity Services Guide



A DocuBrain® Product

https://docubrain.com/

By Prevo Technologies, Inc.

https://prevo.com/

TM

DocuBrain® TechDoc SharePoint Business Connectivity Services Guide

By Prevo Technologies, Inc.

Published by Prevo Technologies, Inc., 1111 Keener Rd, Seymour, TN, 37865.

Table of Contents

Revision History

Revision	Date	Comments
10	02/12/2020	Initial TechDoc 10 SharePoint BCS Guide Release

1. Introduction

TechDoc has had a fully featured SOAP web service stack for quite some time now. The original SOAP service stack was based on Username Token security meaning a TechDoc username and password. While this works well in many cases, it does not work in conjunction with Single Sign-On. To address this issue, TechDoc introduced a full second set of SOAP services that allows authentication using a SAML assertion. Using this second set of services, a client application can pass a user's Single Sign-On token to authenticate and gain access to TechDoc.

After this second set of services was published, it was brought to light that TechDoc was now rather close to being able to interoperate with SharePoint via their Business Connectivity Services (BCS). While BCS is based on SOAP, it does differ quite a lot do to the fact that BCS expects a very particular operation structure and object tree. It was quickly identified that SharePoint could authenticate to TechDoc on behalf of a user with their authentication token using the new SAML secured web service stack, but nearly all of the TechDoc standard operations did not fit the SharePoint BCS mold and therefore could not be used.

Thus, this brings us to the newest offering of TechDoc SOAP services; the TechDoc BCS SOAP services. Unlike the standard SOAP service stacks where there is a separate service for each TechDoc object type i.e. a document service, folder service, etc., all of the operations for the BCS services are unified into a single service in order to be compatible with SharePoint Designer as Designer does not allow crossing from one web service to another when establishing External Content Types (ECTs). While this does make for a rather large service, the user must only consume and configure the service a single time in Designer (when creating the first ECT).

2. Getting Started

In the sections below, we'll cover the steps needed for the initial configuration of TechDoc and SharePoint. Then we'll go over all of the service operations. Finally, we will detail the steps that should be followed as a guideline when importing the service in SharePoint Designer and creating the ECTs for all of the object types.

It is necessary to have prior working experience with SharePoint Designer, an understanding of SOAP services, and a general idea of how External Content Types are created within SharePoint Designer. This guide will not cover the basics of SharePoint or SharePoint Designer.

3. Overview of TechDoc BCS SOAP Service

The TechDoc BCS SOAP service is a single, all-inclusive web service that features all of the operations available for all of the TechDoc object types that are currently supported. The TechDoc BCS SOAP service fully supports TechDoc documents, folders and generations and all of the supporting objects these require, such as TechDoc organizations, users, document categories, etc. While every operation of the standard document, folder and generation service is present, it is important to note that only the standard Create, Read, Update and Delete functions (CRUD) are supported directly within SharePoint BCS/Designer.

BCS/Designer has a very small and strict support for operation types namely, only Created, Delete, Update, Read Item and Read List operations are supported directly. While all of the other operations may be used as well, custom .Net code is required to define the operation. However, this makes sense as SharePoint BCS has no concept of TechDoc, its structure, how it's operations work, etc. It is up to the client to create and develop the custom .Net code they need to connect their specific SharePoint environment to these other TechDoc BCS SOAP operations. As every ECT created within SharePoint Designer varies greatly, it is not possible to create a one size fits all .Net solution that is guaranteed to work in every environment. For example, the BCS XML code generated by Designer when creating an ECT for a TechDoc Folder will not be the same from one SharePoint environment to another; in fact, it is guaranteed not to be in order to avoid collisions.

As stated, currently the main three staple ECTs that may be created in Designer are for TechDoc documents, folders and generations. For these three ECTs, you will find a Create, Update, Delete, Read Item and Read List operation. It's important to note that another unfortunate shortcoming of SharePoint Designer/BCS is that it does not directly support working with binary and as such, the creation of a document is limited to creating a non-resident document in TechDoc that does not contain the actual document.

In order to support binary, the client must create a custom SharePoint solution to handle calling the other document create operation that does take binary. We will not be covering the development of a SharePoint solution in this guide as there are a vast number of tutorials on this topic available directly from Microsoft. If binary interoperability is needed or the use of TechDoc specific operations (non-CRUD based operations such as associating access on a document, releasing a document, etc.), the client will need to develop their own custom SharePoint solution.

4. TechDoc and SharePoint configuration

Before we can begin to use TechDoc SOAP services from SharePoint, we must first configure the trust between TechDoc and SharePoint. When SharePoint connects to TechDoc via web services it will be using claims based authentication on behalf of the user accessing the web services. When the TechDoc BCS web service is ingested in Designer, claims based authentication must be enabled. In doing so, when a user on a SharePoint site attempts to access say a SharePoint List that is populated by a TechDoc document ECT, SharePoint BCS will use that user's claims when calling to TechDoc to access all of the documents for that list.

In order to establish a trust in TechDoc, you must first have a SAML authenticator set up. These days nearly every TechDoc Document Manager out there is leveraging Single Sign-On so most already have a SAML authenticator set up. If your target Document Manager does not yet have a SAML authenticator set up, you'll want to go and do this first before proceeding any further. For more information about this, you can visit the Document Manager -> Admin -> Authenticators page for help with setting up a SAML authenticator.

The first thing you'll need to do is to obtain the message signing and assertion signing certificates for your SharePoint's Secure Token Service (STS). This can be done is several different ways, but the easiest way is to get on the SharePoint farm machine itself and open MMC. Once MMC is open, add the Certificates snap-in for the local computer account and then click OK. Now in the left column, expand the Certificates node and then expand the SharePoint node. Finally, click the Certificates folder under SharePoint. In the column on the right, you should see all of the SharePoint Security Token Service certificates. You'll want to export the certificates labeled as "SharePoint Security Token Service" and "SharePoint Security Token Service Encryption" to a base 64 encoded certificate file. Make sure you do not include the private key.

Now that you have both of the SharePoint STS certificates, copy them over onto the TechDoc Document Manager's machine to a temporary location such as the desktop. Open the File Explorer and navigate to the TechDoc directory's *etc* folder; typically, this is *C:\TechDoc\etc* or *D:\TechDoc\etc*. Once in the TechDoc *etc* folder, create a new file called *samlTrustedClientSHAREPOINT_INSTANCE_NAME.ini* replacing *SHAREPOINT_INSTANCE_NAME* with the name of the SharePoint instance that will be calling TechDoc. Now open this file in a text editor so it can be edited.

Once the configuration file is open in a text editor, and paste in the blue text below:

```
[SamlClientConfiguration]
MessageSigningCertificate=MIIERkjsdackjk4kj45jhvvjhv23jhv3hvbjsdkjsdckefwef
AssertionSigningCertificate= MIIERkjsdackjk4kj45jhvvjhv23jhv3hvbjsdkjsdckefwef
```

NameIDFormat=urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName
NameIDRegex=MY_DOMAIN\\(.*)

[AttributesToExternalGroupIdentifiers]
Name1=groupsid
Name2=denyonlysid
Name3=role

[IntermediateCertificates]

In the first section "SamlClientConfiguration", you'll want to replace the values for MessageSigningCertificate and AssertionSigningCertificate with the base-64 encoded SharePoint STS certificate. Typically, the SharePoint STS will use the same certificate for both the assertion signing and message signing. If assertion encryption is enabled on the SharePoint side, you'll want to use the SharePoint STS Encryption certificate for the AssertionSigningCertificate. Note: in the example above, the MessageSigningCertificate and AssertionSigningCertificate have been shorten; real base-64 encoded certificates should be much longer.

Now that the certificates are correct, you'll next want to turn your attention to the NameIDRegex field. This field can be omitted if the subject name being sent in the SharePoint claim will match letter for letter to the TechDoc user's SAML authenticator username. Many times, however, most SharePoint instances will return the name ID with the Windows domain\username and this will not work as TechDoc expects to receive only a SAML standard username. In order to accept this nonstandard username format, you can opt to use a NameIDRegex regular expression. In the example above you'll see that we have MY_DOMAIN\\(.*) specified. This says to strip MY_DOMAIN\ and keep everything after it. Using this expression, we can strip away all but username so it will map to the TechDoc user's SAML username in TechDoc. The NameID configuration will depend solely on how your SharePoint instance is configured. We'll talk a bit more about this in the next little bit when we configure the TechDoc SAML authenticator.

In the next section "AttributesToExternalGroupIdentifiers", you will see multiple SAML attribute to TechDoc external group mappings. This says for every groupsid, denyonlysid, and role encountered in a SAML assertion, attempt to map those to a TechDoc external group. This adds some additional flexibility to gain access to TechDoc for users that do not have a TechDoc user account. For example, say that the user MY_DOMAIN\johnsmith attempts to view a list of TechDoc documents on a SharePoint site. Using our logic above, we will strip MY_DOMAIN\ leaving "johnsmith" as the username. SharePoint will be attempting to list documents from TechDoc using johnsmith but johnsmith does not have a TechDoc account. Normally, he would not be able to access anything without an account.

However, it just so happens that johnsmith's SAML assertion includes some Active Directory groupsids that TechDoc knows about; one of these groupsids is the Acme Employee's group. Say for instance this TechDoc Document Manager has an external group configured for the Acme Employee groupid and this external group has access to the documents johnsmith is trying to access. Because we listed groupsid as one of the SAML attributes we want to parse to a potential TechDoc external group AND johnsmith's SAML assertion contained the ACME Employee's groupsid AND TechDoc has an external group configured for the ACME Employees group AND finally the ACME Employee's group has read access to the documents johnsmith is trying to retrieve, johnsmith is still able to view these documents because while he does not have a TechDoc user account he does still have access by way of an external group.

You'll want to review each of the items listed under AttributesToExternalGroupIdentifiers to see if you want to include any of these or if there are any others you wish to include in addition to these. Remember that these alone will not automatically grant the user access in TechDoc. Each groupsid will need to be first set up in a TechDoc group and then that group must be given access to documents. Only then will each groupsid actually give non-TechDoc users access to TechDoc resources. Each item listed in AttributesToExternalGroupIdentifiers is prefixed with Name a number then an equal sign. For example, in the list above we have Name1, Name2 and Name3. If we wanted to add an additional SAML attribute we would add Name4= and then the name of the SAML attribute. If we wanted to remove an attribute, we would reorder the list if needed so that it contained just Name1 and Name2.

Finally, in the last section "IntermediateCertificates", you'll want to add any intermediate certificates your SharePoint STS and optionally SharePoint STS Encryption certificate may need to validate all the way back to the root. You can double click each of these certificate files and view the hierarchy on the TechDoc machine to see if any intermediate certificates are missing on the TechDoc machine. If you do encounter any certs that need to be added, jump back over on the SharePoint machine and export each of those intermediate certs to base-64 cert files like you did previously with the others. Next include each of those certs in the IntermediateCertificates section using a number scheme like:

```
[IntermediateCertificates]
Cert1=BASE_64_CERT_HERE
Cert2=BASE_64_CERT_HERE
(etc.)
```

The configuration file should now be complete. Save the file and then log into the TechDoc DM with an admin account. Navigate to the Admin screen and click Authenticators under the Show… list. Locate your Document Manager's main SAML authenticator that you will be connecting through with SharePoint and then click

modify. On the Service Data line, add a -c option immediate followed by the name of the configuration file like:

-c*samlTrustedClientSHAREPOINT_INSTANCE_NAME.ini*

Being sure to adjust the filename above to match the file name you chose. It's important to note that the full file path does not need to be specified, only the filename. TechDoc knows to look in its own *etc* folder. Also, make sure there is not a space between -c and the name of the file.

Once you have configured this authenticator switch, enter a reason and then click OK. Congratulations, your TechDoc instance now trusts your SharePoint instance and is ready to receive SAML based Active Directory based claims!

4.1. *Document Manager System Properties*

In this section, we'll cover all of the TechDoc DM system properties and *etc* file settings that affect the functionality of the TechDoc SharePoint BCS service. The only TechDoc DM system property affecting SharePoint BCS is:

SharePointBCSDefaultMaxResults

This system property can be used to control the amount of results returned from the various read list based web service operations. For example, if Document_ReadList is invoked with no parameters, the TechDoc DM would attempt to send the metadata for every single document in the system (that the user has read access to, of course)! Obviously, this is not ideal and can consume way too many resources in TechDoc, SharePoint, and the IT infrastructure supporting the two. To deal with this, the SharePoint BCS Default Max Results system property can be used to limit the default number of results any listing based method should ever return. By default, this property is set to 1000. If for some reason it's needed and the ramifications are well understood, this property can be set to 0 to allow an unlimited number of results. Take care when using zero.

4.2. *Document Manager ETC File*

In addition to the above-mentioned system property, it is also possible to enable logging for the TechDoc SharePoint BCS Service. This logging mechanism can be turned on by editing the *dm.ini* and adding the following section:

[logging]
TechDocBcsService=1

The value of 1 signifies that logging should be enabled whereas a value of 0 indicates logging should be disabled. Additionally, if the configuration parameter is not specified at all, the logging mechanism defaults to being turned off.

The logging mechanism on the service can be a huge help to debug issues when working with BCS. The most common issue we've come across when working with BCS is what input parameters it passes to the various TechDoc service calls on your behalf. While most of the time, it passes letter for letter what you specify in an input field or search filter, at times it can change the user's input based on how that input field or search filter is set up. The logging mechanism will log the operation name, each of the parameters and the exact value it received for it. When enabled, any logged output can be viewed in the TechDoc system log.

5. Operations Directly Supported

In this section, we'll go over the operations that are directly supported within SharePoint Designer that require no custom .Net code. As stated above, this functionality will be limited only to standard Create, Read, Update and Delete (CRUD) operations with a few exceptions. SharePoint Designer only supports CRUD operations and does not support any of TechDoc's custom operations such as associating access on a document, releasing a document, etc. This does however make sense as SharePoint has no concept of the innerworkings of TechDoc and would not know or understand any of these custom operations. Additionally, binary is not directly supported within Designer so the creation of documents is limited to non-resident documents only.

Below is a list of the main three object type that ECTs will be created for and their corresponding CRUD operations that are supported without any additional custom code. All of these ECTs are/will be uniquely identified by their ID.

1. Document ECT
    - *Document_Create* – Used to create a non-resident document.
    - *Document_ReadItemByID* – Used to read a single document.
    - *Document_ReadList* – Used to read a list of or search for documents.
    - *Document_Update* – Used to update a document.
    - *Document_Delete* – Used to delete a document.
2. Folder ECT
    - *Folder_Create* – Used to create a folder.
    - *Folder_ReadItemByID* – Used to read a single folder.
    - *Folder_ReadList* – Used to read a list of or search for folders.
    - *Folder_Update* – Used to update a folder.
    - *Folder_Delete* – Used to delete a folder.
3. Generation ECT
    - *Generation_ReadItemByID* – Used to read a single generation.
    - *Generation_ReadList* – Used to read a list of or search for generations.

- *Generation_Update* – Used to update a generation.
- *Generation_Delete* – Used to delete a generation.

Also, there are numerous other operations that are directly supported that can be used to define the supporting objects ECTs for the document, folder and generation ECT. For example, a TechDoc document has associated to it a document category, document type, owner, organization, etc.

In the list below, you will find the minimum operations needed to establish ECTs for all of the supporting objects. Keep in mind, most all of these object types will only have a read item and read list operation as the creation of most of them are an Admin only function in TechDoc and thus are not available for use through SOAP. Most of these objects are all identified by a unique but user-friendly identifier. The idea here is that when say the document ECT is created, all of its fields are user readable and are not just a bunch of ID fields. In this way, if it is not desired to create all of these supporting ECTs, when a single document, folder or generation is viewed all its fields will be user readable (as opposed to a bunch of IDs). Furthermore, since most of these supporting ECTs do not support the create and update operations (as they are admin only), there is no need for a numeric ID.

1. Document Category ECT (Used by the Document ECT)
    - *DocumentCategory_ReadItemByAbbreviation* – Used to read a single document category.
    - *DocumentCategory_ReadList* – Used to list all of the document categories in TechDoc.
2. Document Keyword ECT (Used by the Document ECT)
    - *DocumentKeyword_Create* – Used to create a document keyword on a document.
    - *DocumentKeyword_ReadItemByID* – Used to read a single document keyword.
    - *DocumentKeyword_ReadList* – Used to read a list of or search for document keywords.
    - *DocumentKeyword_Update* – Used to update a document keyword.
    - *DocumentKeyword_Delete* – Used to delete a document keyword.

3. Document Read Access ECT (Used by the Document ECT)

    - *DocumentReadAccess_ReadItemByName* – Used to read a single document access value.

    - *DocumentReadAccess_ReadList* – Used to list all of the document read access values in TechDoc.

4. Document Type ECT (Used by the Document ECT)

    - *DocumentType_ReadItemByAbbreviation* – Used to read a single document type.

    - *DocumentType_ReadList* – Used to list all of the document types in TechDoc.

5. Document Web Search ECT (Used by the Document ECT)

    - *DocumentWebSearch_ReadItemByName* – Used to read a single document web search value.

    - *DocumentWebSearch_ReadList* – Used to list all of the document web search values in TechDoc.

6. Folder Read Access ECT (Used by the Folder ECT)

    - *FolderReadAccess_ReadItemByName* – Used to read a single folder read access value.

    - *FolderReadAccess_ReadList* – Used to list all of the folder read access values in TechDoc.

7. Group ECT

    - *GroupReadItemByName* – Used to read a single group.

    - *GroupReadList* – Used to list/search all of the groups in TechDoc.

8. Keyword ECT (Used by the Document Keyword ECT)

    - *Keyword_ReadItemByName* – Used to read a single keyword.

    - *Keyword_ReadList* – Used to list/search all of the keywords in TechDoc.

9. Organization ECT (Used by the Document and Folder ECT)

    - *Organization_ReadItemByAbbreviation* – Used to read a single organization.

    - *Organization_ReadList* – Used to list/search all of the organizations in TechDoc.

10. RMA File Plan ECT (Used by the Folder ECT)

    - *RMAFilePlan_ReadItemByName* – Used to read a single RMA file plan.

    - *RMAFilePlan_ReadList* – Used to list/search all of the RMA file plans in TechDoc.

11. User ECT (Used by the Document and Folder ECT)

    - *User_ReadItemByUsername* – Used to read a single user.

    - *User_ReadList* Used to list/search all of the users in TechDoc.

That wraps up the list of all the supporting ECTs needed by documents, folders and generations. You'll notice in the list above, most all of the ECTs are for either document or folder as generation is mainly reliant on its parent document.

Though it is not necessary to create an ECT for all of the supporting objects listed above, it is highly recommended as without them a SharePoint user will have to manually enter IDs for all of these fields as there will not be an association established. If there is not a supporting ECT and association, SharePoint cannot provide the nice lookup list functionality on a field that lets the user navigate through a look up directory for all of these fields. Thus, it's highly recommended that all of these be created as well following the instructions later on in this document.

6. Operations That Require Custom Code

In this section, we'll go over all of the other TechDoc specific (non-standard CRUD) functionality. All of these operations are ones that do not fit the CRUD mold and are not supported directly by Designer or BCS and require custom .Net code to call. Again, we'll go through all of these operations sorted by the ECT they belong to.

1. Document ECT

    - *Document_AssociateAccess* – This operation is used to set the user access on a specific document. This access specifies what users or groups can view the document, modify the document, delete the document, etc.

    - *Document_AssociateCommenters* – This operation is used to specify which users and groups are allowed to specify comments on the target document.

    - *Document_AssociateDistribution* – This operation is used to specify the users and groups that will receive email when a document has been released for distribution.

    - *Document_AssociateNotification* – This operation is used to specify the users and groups that will receive email any time a change is made on a document. For example, if a document is modified, replaced, etc.

    - *Document_CreateResidentDocument* – This operation is the same as the other *Document_Create* listed above except this operation also supports receiving the binary file for the document (thus making it a resident document). Binary is not directly supported in BCS without custom .Net. BCS requires that all binary data be handled in a custom .Net code shim to allow BCS to communicate binary to and from standard SOAP binary streams. Specifically, any operations that return or take as a parameter, an XML mime type of application/octet-stream.

    - *Document_DeleteComment* – This operation can be used to delete a specific comment on a document if needed.

    - *Document_ExistsByDocNumber* – This operation is a convenience method that allows for easy checking to see if a document exists in TechDoc using its document number.

    - *Document_ExistsByID* – This operation is another convenience method that allows for easy checking to see if a document exists in TechDoc using its ID.

- *Document_Fetch* – This operation is used to perform a fetch (download) of a document from TechDoc. As with the create resident document above, BCS doesn't support binary directly so downloading a file from TechDoc requires custom .Net code as well.

- *Document_GetGenerations* – This is a simple operation to find all the generations of a specific document. The IDs returned are generation IDs and may be passed directly to a *Generation_ReadItemByID* call.

- *Document_Release* - This operation performs a release of a document in TechDoc. In TechDoc, when a document is released, it means that the document is now ready for official use and is usually sent over to one or more TechDoc Search Managers to make for easy searching.

- *Document_Replace* – This operation is used to replace a document in TechDoc with another document. This operation can either replace the existing generation with a new replacement generation, or alternatively just create a new additional generation with the file specified.

- *Document_Reserve* – The operation is used to reserve a document in TechDoc; in some other systems, this is also known as a check out. Typically, a user should first reserve a document before they begin making changes to ensure that two people aren't trying to make changes to the same document.

- *Document_ShowAssociations* – This operation can be called to target a specific document and retrieve details about what users and groups have what kinds of access to the document.

- *Document_Unrelease* – This operation is similar to the release operation listed above. On an unrelease, the target document is retracted from any of the Search Managers where it is located and the corresponding generation loses its released status.

- *Document_Unreserve* – This operation is used to undo a prior reserve operation. On an unreserve operation, the document reservation is removed without replacing or adding a generation to the document freeing the document to now be reserved by someone else.

2. Folder ECT

    - *Folder_AssociateAccess* – This operation is used to set the user access on a specific folder. This access specifies what users or groups can view the folder, modify the folder, delete the folder, etc.

- *Folder_CreateFolderTree* – This operation functions just like a normal *Folder_Create* except it will create all of the missing folders in the path specified. For example, say only a root cabinet of /Test exists and you perform a *Folder_CreateFolderTree* with a path of */Test/Folder1/Folder2/DestinationFolder*. Typically, with a normal *Folder_Create* this would fail as both */Test/Folder1* and */Test/Folder1/Folder2* do not already exist but it succeeds with *Folder_CreateFolderTree* as it will create each missing intermediate folder before finally creating the lowest level target folder.

- *Folder_ExistsByID* – This operation is another convenience method that allows for easy checking to see if a folder exists in TechDoc using its ID.

- *Folder_ExistsByPath* – This operation is another convenience method that allows for easy checking to see if a folder exists in TechDoc using its full path.

- *Folder_GetChildDocuments* – This operation can be used to list all of the documents in a specific folder.

- *Folder_GetChildFolders* – This operation can be used to list all of the child folders in a specific folder.

- *Folder_MoveContents* – This operation can be used to move the contents of one folder (both child documents and child folders) to another folder.

3. Generation ECT

    - *Generation_CreateRendition* – This operation is used to create a rendition for the generation specified; a rendition is a rendered PDF of the original file. Normally, the system will create a rendition when a document is released but sometimes it is desirable for the user to put in their own custom created rendition.

    - *Generation_ExistsByGenNumber* – This operation is a convenience method that allows for easy checking to see if a generation exists in TechDoc using its generation number and parent document number.

    - *Generation_ExistsByID* – This operation is another convenience method that allows for easy checking to see if a generation exists in TechDoc using its ID.

    - *Generation_ReplaceRendition* – This operation is used to replace an already existing rendered PDF on a specific generation with a new one.

Normally, this is unnecessary but occasionally it is desirable for the user to put in their own custom created rendition.

- *Generation_ResubmitRendition* – This operation is used to ask TechDoc to attempt to regenerate a rendition for the generation specified.

- *Generation_UpdateReleaseDate* – This operation is used to update the release date of the generation specified.

7. Building ECTs in Designer

In this section, we will go over the steps needed to build each of the External Content Types (ECTs) in SharePoint Designer for each of the TechDoc objects we'll be working with. Up to this point, it's expected that you have already configured the TechDoc SharePoint trust from the previous section and reviewed all of the operations that you may use directly from Designer and those you may not.

This guide is based on SharePoint 2013 and SharePoint Designer 2013. If a newer or older version of SharePoint and Designer are being used, be careful to review everything that is being done below to make sure it will work correctly in your version of SharePoint. Please note, this section should only be used as a set of basic guidelines to establish the basic TechDoc object structure in Designer. Most implementations of this ECT object structure in test and production are likely to be highly customized to the user's need. As such, we'll just go over the very basics needed to set up each ECT.

7.1. *Supporting ECTs*

Launch SharePoint Designer 2013 and connect to the SharePoint instance you'll be working with. We'll first start by building all of the supporting ECTs this way when we go to build the main ECTs (documents and folders), we'll be able to connect all of our supporting ECTs.

In the following blocks below, you'll find a block for each ECT. Each block will have the Name, Display, object field settings, etc. The first ECT will require you to make a connection to TechDoc, after this first ECT you should reuse the connection for all of the subsequent ECTs. We'll create the first supporting ECT together detailing each and every step. Start by clicking External Content Types in the left column and then click the new External Content Type button on the ribbon bar.

7.1.1. Document Categories

- Name – *TechDoc Document Category*
- Display Name – *documentCategory*

For this ECT, you'll want to map:

- *DocumentCategory_ReadItemByAbbreviation* to the Read Item operation.
- *DocumentCategory_ReadList* to the Read List operation.

Click the Name field and enter "TechDoc Document Category" without the quotes and then click Display Name and enter "documentCategory" without the quotes. Next click the "Click here to discover external data…" link under External System. Click the Add Connection button, select WCF service from the Data Source Type drop down and then click OK.

On the WCF Connection dialog, you'll want to enter in the Service Metadata URL field:

https://*TECHDOC_DM_HOST*/service/soap/bcs/TechDocBcsService?wsdl

replacing *TECHDOC_DM_HOST* with the fully qualified host name of your TechDoc DM. HTTPS is mandatory; the TechDoc BCS service will not work over clear text HTTP.

Make sure the Metadata Connection Mode is set to WSDL and enter in the Service Endpoint URL field:

https://*TECHDOC_DM_HOST*/service/soap/bcs/TechDocBcsService

Again, replacing *TECHDOC_DM_HOST* with the proper host name for your TechDoc DM. Optionally, enter a Name and any proxy information needed (note we do not test against a proxy server, but this should not impact the usage of the service). Under WCF Service Authentication Settings, make sure that "Connect with User's Identity" is selected as well as "Use the same connection settings for metadata retrieval". Finally click OK to finish creating this connection.

Next, we'll create the Read Item and Read List operations for Document Category. Expand the newly created connection in the left column, expand Web Methods, locate *DocumentCategory_ReadItemByAbbreviation*, right click it, and select New Read Item Operation. On the Read Item dialog, click the Next button. Select *documentCategoryAbbreviation* in the left-hand input parameters column of the, check the "Map to Identifier" box, and then click the Next button. Again, select *documentCategoryAbbreviation* in the left-hand output parameters column, check the "Map to Identifier" box, and then click Finish.

Now we'll create the Read List operation. In the left-hand column under Web Methods, right click on *DocumentCategory_ReadList* and select New Read List Operation. On the Read List dialog, click Next. Click *documentCategoryAbbreviation* in the left-hand input parameters column and then click the "Click to Add" link under Filter. Select New Filter and enter "Document Category Abbreviation" without the quotes. Select Comparison for the Filter Type, Equals for the Operation, and then click OK. We've now created our first search filter. This will allow users to search specifically by document category abbreviation when attempting to locate a document category in a list. On the Read List dialog, click Next to continue. Select *documentCategoryAbbreviation* in the return

parameter column and check the "Map to Identifier" checkbox. Finally, click Finish to complete the read list operation.

We've now created a Read Item operation that describes the basic fields of a document category as well as a Read List operation that allows locating one/many/all of the document categories as well as searching through those categories by abbreviation. The last thing we need to do is to configure the web service connection to use Claims based authentication (this cannot be done until at least one web service operation has been created hence why we had to wait until now to configure it). To enable claims based authentication, click the documentCategory breadcrumb to jump back to the top level of the documentCategory ECT. Now under the External System field, you should see the name of the TechDoc BCS service you created just a few minutes earlier; click this link. On the Connection Properties window, check the box next to "Use claims based authentication" and then click OK. This step is very important and should not be missed. The TechDoc web service will not grant access without the user's claims.

At this point, you should save this ECT by clicking the Save icon on the top bar. Once the ECT has been saved off to SharePoint, you may optionally create a SharePoint List for this new ECT. It is suggested that you create a SharePoint List for every ECT created during development to test functionality. Later on, these lists can be deleted if they are no longer needed. To create a SharePoint List, make sure you are viewing the document category ECT (make sure the documentCategory ECT tab is selected) and then click the "Create Lists & Form" button on the top ribbon bar. Enter a name for the list, something like TechDoc Document Categories and then click the OK button. Again, click the Save icon on the top menu bar to save off all of your changes.

You should now be able to log onto your SharePoint site and navigate to the list. Verify that you can view all of the document categories in the TechDoc DM you are connected to. If you cannot view the document categories or run into a SharePoint exception, review all of the steps above paying close attention to the web service configuration. Again, make sure claims based authentication is enabled because the web service connection will not be granted access without those claims. If you are still not able to connect, you may need to get with a SharePoint support person and review the SharePoint logs to see if BCS is running into any issues. The first connection is always the trickiest, but as long as your SharePoint instance is vanilla and not to highly customized, everything should work following the steps above. Do not proceed further until this basic functionality test is working.

After you have verified that you can view all of the document categories in TechDoc, you may proceed on through the following blocks below creating an ECT and corresponding SharePoint list for each block.

7.1.2. Document Read Access Values

- Name – *TechDoc Document Read Access Values*
- Display Name – *readAccess*

For this ECT, you'll want to map:

- *DocumentReadAccess_ReadItemByName* to the Read Item operation.
- *DocumentReadAccess_ReadList* to the Read List operation.

7.1.3. Document Types

- Name – *TechDoc Document Types*
- Display Name – *doctype*

For this ECT, you'll want to map:

- *DocumentType_ReadItemByAbbreviation* to the Read Item operation.
- *DocumentType_ReadList* to the Read List operation.

7.1.4. Document Web Search Values

- Name – *TechDoc Document Web Search Values*
- Display Name – *webSearch*

For this ECT, you'll want to map:

- *DocumentWebSearch_ReadItemByName* to the Read Item operation.
- *DocumentWebSearch_ReadList* to the Read List operation.

7.1.5. Folder Read Access Values

- Name – *TechDoc Folder Read Access Values*
- Display Name – *readAccess*

For this ECT, you'll want to map:

- *FolderReadAccess_ReadItemByName* to the Read Item operation.
- *FolderReadAccess_ReadList* to the Read List operation.

7.1.6. Generations

Generation is more of a prominent object type in TechDoc however, the direct creation of a generation is not supported to due BCS's lack of support for binary. Additionally, all of the other generation operations are custom and require .Net code. Because of this, we'll be working with generation here solely as a supporting object with just a read item and read list operation.

- Name – *TechDoc Generations*
- Display Name – *generation*

For this ECT, you'll want to map:

- *Generation_ReadItemByID* to the Read Item operation.
- *Generation_ReadList* to the Read List operation.

7.1.7. Organizations

- Name – *TechDoc Organizations*
- Display Name – *organization*

For this ECT, you'll want to map:

- *Organization_ReadItemByAbbreviation* to the Read Item operation.
- *Organization_ReadList* to the Read List operation.

7.1.8. RMA File Plans

- Name – *TechDoc RMA File Plans*

- Display Name – *rmaFilePlan*

For this ECT, you'll want to map:

- *RmaFilePlan_ReadItemByAbbreviation* to the Read Item operation.
- *RmaFilePlan_ReadList* to the Read List operation.

7.1.9. **Users**

- Name – *TechDoc Users*
- Display Name – *user*

For this ECT, you'll want to map:

- *User_ReadItemByUsername* to the Read Item operation.
- *User_ReadList* to the Read List operation.

7.2. *Main ECTs*

Now that all of the supporting ECTs have been built, we can focus on building the main ECTs for documents, folders and document keywords. In the next few blocks below, we'll go over the main ECTs in the same fashion as we have the ECTs above, but we'll detail a bit more on the more complex operations like create and modify.

7.2.1. **Documents**

- Name – *TechDoc Documents*
- Display Name – *document*

For this ECT, you'll want to map:

- *Document_ReadItemById* to the Read Item operation.
- *Document_ReadList* to the Read List operation.
- *Document_Create* to the Create operation.
- *Document_Update* to the Update operation.
- *Document_Delete* to the Delete operation.
- Create a reverse association to the document category ECT joined on the *documentCategoryAbbreviation* field to the *Document_ReadItemById* operation.
- Create a reverse association to the document type ECT joined on the *documentTypeAbbreviation* field to the *Document_ReadItemById* operation.
- Create a reverse association to the organization ECT joined on the *organizationAbbreviation* field to the *Document_ReadItemById* operation.
- Create a reverse association to the document web search values ECT joined on the *webSearch* field to the *Document_ReadItemById* operation.
- Create a reverse association to the document read access values ECT joined on the *readAccess* field to the *Document_ReadItemById* operation.

The document ECT is quite a bit more complex than the other supporting ECTs. Specifically, there are several fields values than must be marked read only and some that must be unchecked. Start by creating a new ECT in Designer with a Name of "TechDoc Documents" and a Display Name of "document" both without the quotes. Next, click the External System link so that you may begin adding operations.

We'll first start by creating the read item operation. Right click on *Document_ReadItemByID* and select New Read Item Operation. The on Read Item dialog, click the Next button. On the input parameters screen, select *docID,* check the "Map to Identifier" box and then click the Next button. On the return parameters screen, you will see many parameters listed. Let's first start by unchecking the boxes next to the following parameters so that we can ignore them completely:

- Uncheck the box next to *createdSpecified*.
- Uncheck the box next to *modifiedSpecified*.
- Uncheck the box next to *releasedSpecified*.

SharePoint BCS cannot handle standard null date/time values, but TechDoc does and BCS must support this. For whatever reason, BCS creates a -Specified field for each date time value it encounters to use as a bit of a crutch to help deal with this. Unfortunately, it then expects that the client (in our case TechDoc) will know about these imaginary fields and expect the client to take these as parameters. TechDoc of course knows nothing about these and as such does not offer or accept them. This in turns causes issues in SharePoint BCS if the fields are left checked as the ECT's object now has fields the backend service ignores upon create, update, etc. The best solution is to uncheck these fields and any other imaginary fields you see SharePoint BCS/Designer create.

Next, we'll need to go through the whole list of parameters and mark the read only parameters as read only so that BCS will know the parameters are for display only and should not be editable when the user is creating or updating a document. For the following parameters, select the parameter in the left-hand column and then check the Read-Only box in the right-hand column:

- *created*
- *inReview*
- *linkExploreDocument*
- *linkFetchLatestGeneration*
- *linkFetchLatestRevision*

- *modified*
- *owner*
- *parentFolderID*
- *released*
- *reservedBy*
- *resident*
- *revision*
- *uuid*

After marking all of the above fields as read only, click the Finish button.

Now that we have established the read item operation, we must create a read list operation. In the left-hand column, right click the *Document_ReadList* and then click New Read List Operation. On the Read List dialog, click the Next button. On the input parameters screen, you are free to create input search filters for any of the parameters you see listed. We suggest that you create at a minimum an Equals Comparison filter for both *docNumber* and *parentFolderPath*. This will let users search for documents by both doc number and their parent folder. You are free to create additional filters if needed. Once the filters have been created, click the Next button. Again, following the same steps above, mark the *docID* as the identifier, uncheck the same three fields and finally mark the same fields as read-only. Once you have done all of this, click the Finish button.

Moving on, we are now ready to create the Create Operation. Right click *Document_Create* and select New Create Operation. On the Create dialog click Next. On the input parameters screen, unselect the following parameters:

- *created*
- *inReview*
- *modified*
- *owner*
- *parentFolderID*
- *released*

- *reservedBy*
- *resident*
- *revision*
- *uuid*

Unchecking all of these input parameters will help guide SharePoint to not display them as input parameters when creating a new document. After all of these parameters are unchecked, click the Next button. On the return parameters screen, expand *Document_Create*, click *docID*, and then click "Map to Identifier". Finally, click the Finish button.

At this point, you should have a read item, read list and create operation listed in the External Content Type Operations box. We now want to create the update operation. Right click the *Document_Update* operation in the left-hand column and then click New Update Operation. On the Update dialog, click the Next button. Select the *docID* parameter and mark it as the identifier. Next uncheck the following parameters:

- *created*
- *inReview*
- *modified*
- *owner*
- *parentFolderID*
- *released*
- *reservedBy*
- *resident*
- *revision*
- *uuid*

Again, this will instruct SharePoint to not display an input field for these parameters when performing a document update. After the *docID* has been marked as the identifier and the appropriate fields are unchecked, click the Finish button.

Now we'll create the delete operation. Right click *Document_Delete* and then click New Delete Operation. On the Delete dialog, click the Next button. Select the *docID* parameter as mark it as the identifier and then click the Finish button.

At this point, you now have all of the CRUD operations built and could begin using the ECT however, you would not have the lookup list abilities for several of the document ECT fields where it's quite nice to have. To add these lookup lists to the ECT, you must simply create reverse associations out to the supporting ECTs that each of the fields correspond to.

Let's start by creating the reverse association to the document category ECT. Right click *Document_ReadItemById* and then select New Reverse Association. On the Reserve Association dialog, click the Browse… button and then select the TechDoc Document Categories ETC. Next, enter an Association Name and Display Name of Document Category Association. Next, select *documentCategoryAbbreviation* in the field drop down box to link the document ECT's *documentCategoryAbbreviation* with the *documentCategoryAbbreviation* field in the document category ECT and then click the Next button. On the input parameters screen, select *docID*, mark it as the identifier, and then click the Next button. On the return parameters screen, select *documentCategoryAbbreviation* and then click "Map to Identifier". The remaining parameters on this screen may be left as they are as BCS will only use the abbreviation field during association. Click the Finish button to complete the reverse association.

You've now created your first reverse association. Now when creating or modifying a document you no longer have to key in a document category abbreviation by hand. SharePoint will display a nice lookup list and live fetch the available document categories from TechDoc for you.

We now want to create reverse associations for all of the other applicable fields on the document ECT. Following the same steps, create a reverse association on *Document_ReadItemById* to the Document Type ECT, Organization ECT, Document Web Search ECT, and Document Read Access ECT joined on the *doctypeAbbreviation, organizationAbbreviation*, *webSearch,* and *readAccess* fields, respectively. Make sure to name the associations in the same manner as the first and to mark the *docID* input parameter as an identifier and the corresponding output field *(doctypeAbbreviation, organizationAbbreviation*, *webSearch,* or *readAccess*) as an identifier.

You should now have a read item, read list, create, update, and delete operation defined as well as reverse associations to the document category, document type, organization, document web search, and document read access ECTs. Once you verify that you have all of these defined, click the Save icon to propagate this ECT into SharePoint and then create a SharePoint List for the document ECT. This completes the basic configuration of the document ECT in SharePoint Designer. You may further

customize this ECT by adding additional search fields to Read List, customizing the display of the fields, etc.

Please note that while all of this customization is available, it is rather easy to break the complex object model with the smallest change due to its complexity. It advised to pre-plan out the additions and modifications that will be made and potentially snapshot your SharePoint instance beforehand. This way, should a small change break the BCS document ECT model, you have a quick means of recovery to get everything back up and working. It can be a bear to try and figure out what's broken after the fact and fix it after a bad addition or modification.

7.2.2. **Document Keywords**

- Name – *TechDoc Document Keywords*
- Display Name – *documentKeyword*

For this ECT, you'll want to map:

- *DocumentKeyword_ReadItemByID* to the Read Item operation.
- *DocumentKeyword_ReadList* to the Read List operation.
- *DocumentKeyword_Create* to the Create Operation.
- *DocumentKeyword_Update* to the Update operation.
- *DocumentKeyword_Delete* to the Delete operation.
- Create a reverse association to the document ECT joined on the *docID* field to the *DocumentKeyword_ReadItemByID* operation.

Create a new ECT with a name of *TechDoc Document Keywords* and a display name of *documentKeyword*. Following the same steps as you did with the document ECT, first, create a read item and read list operation for the document keyword ECT. When creating the read item and read list operations, be sure to mark *documentKeywordID* as the identifier. During the read list operation, it is advised to create a search filter for all of the input parameters as keywords are heavily used by TechDoc users; if they aren't added now, they'll soon be requested later.

Next, build the operations for create and update. On the input parameters screen, uncheck the *docID* parameter and make sure to mark *documentKeywordID* as the identifier.

Finally, build the delete operation and create the reverse association on the *DocumentKeyword_ReadItemByID* operation against the document ECT joined on the *docID* field. After all of the CRUD operations and reserve association is created, click the Save icon to save the document keyword off to SharePoint. At this point, you should now go ahead and create a SharePoint list as well.

7.2.3. **Folders**

- Name – *TechDoc Folders*
- Display Name – *folder*

For this ECT, you'll want to map:

- *Folder_ReadItemById* to the Read Item operation.
- *Folder_ReadList* to the Read List operation.
- *Folder_Create* to the Create operation.
- *Folder_Update* to the Update operation.
- *Folder_Delete* to the Delete operation.
- Create a reverse association to the organization ECT joined on the *organizationAbbreviation* field to the *Folder_ReadItemById* operation.
- Create a reverse association to the folder read access values ECT joined on the *readAccess* field to the *Folder_ReadItemById* operation.
- Create a reverse association to the RMA file plan ECT joined on the *rmaFilePlanName* field to the *Folder_ReadItemById* operation.

Similar to the document ECT, the folder ECT has several parameter customizations to make to ensure you generate a working folder object model. Create a new ECT with a name of *TechDoc Folders* and a Display Name of *folder*. Then, view the ECT operations so we can begin building each of the operations.

On the Data Source Explorer tab, locate and right click the *Folder_ReadItemById* operation and then select New Read Item Operation. On the Read Item dialog, click the Next button. On the input parameters screen, mark *folderID* as the identifier and then click the Next button. On the return parameters screen, start by unchecking the following fields:

- *createdSpecified*
- *modifiedSpecified*

Next, you'll need to mark the following fields as read only:

- *created*
- *fullPath*
- *modified*
- *owner*
- *rmaRecordSetName*
- *uuid*

After unchecking the fields and marking the others listed as read only, click the Finish button.

Now, right click on Folder_ReadList and select New Read List Operation. On the Read List dialog, click the Next button. On the input parameters screen, it is suggested to at least create a Comparison Equals input filter for parentPath. As before, you may create as many other filters as you lilke. After the filters are created, click the Next button. Again, mark *folderID* as the identifier, unselect the *createdSpecified* and *modifiedSpecified* parameters. Mark *created, fullPath, modified, owner, rmaRecordSetName,* and *uuid* as read only, and then click the Finish button.

Next, we'll create the Create Operation. Right click *Folder_Create* and select New Create Operation. On the Create dialog click the Next button. On the input parameters screen, uncheck the following fields:

- *created*
- *fullPath*
- *modified*
- *owner*
- *rmaRecordSetName*
- *uuid*

This will instruct SharePoint to only display input fields for the other parameters on a folder create operation. Click the Next button. Expand *Folder_Create* and then mark *folderID* as the identifier. Click the Finish button.

Now, right click *Folder_Update* and select New Update Operation. On the Update dialog, click the Next button. On the input parameters screen, mark *folderID* as the identifier and uncheck the following parameters:

- *created*
- *fullPath*
- *modified*
- *owner*
- *rmaRecordSetName*
- *uuid*

Finally, click the Finish button to complete the update operation.

Now, right click *Folder_Delete* and select New Delete Operation. On the Delete dialog, click the Next button. On the input parameters screen, mark *folderID* as the identifier and then click Finish.

At this point, you should have all of the CRUD operations built out but as before with the document ECT, there are a few additional reverse associations to add that will enhance the user experience. Following the same steps as you did with the document ECT, create reverse associations on *Folder_ReadItemByID* against the organization, folder read access and RMA file plan ECTs against the *organizationAbbreviation*, *readAccess*, and *rmaFilePlanName* fields, respectively.

Once you have created these reverse associations, click the Save icon to propagate the folder ECT into SharePoint. Then create a SharePoint List for the folder ECT.

8. Testing TechDoc ECTs

At this point, you should have competed the previous two sections and have a full working set of TechDoc object ECTs in your SharePoint instance. It's suggested that you now log onto the target SharePoint site and test out all of the functionality that is present. Before making any customizations or changes, we recommend that you snapshot your SharePoint instance to provide for an easy roll back of the ECT object structure should any of your changes cause issue. The BCS object structure is quite complex and as such can be fragile. Great care should be taken any time the object structure is being altered as the smallest type or setting change can completely break an ECT and stop it from working entirely in test/production.

Also, it is possible to export all of the TechDoc ECTs into a single SharePoint BCS model file. If any problems should arise, all of the ECTs and Lists can be deleted out using SharePoint Designer and then the backup model file can be re-inserted into the system using Central Admin.

9. TechDoc BCS Overview

In this section. we'll cover all of the complex data objects TechDoc exposes (the things that are later defined as ECTs in SharePoint Designer i.e. the document object, folder object, etc.). After we've covered all of the complex data objects, we'll go over each and every service operation available in the TechDoc BCS service both directly supported by Designer/BCS without custom code and those that require custom code.

9.1. *Complex Data Objects*

Listed below are each of the complex data objects that are to be defined as ECTs in SharePoint Designer. Each object will detail its general purpose and the usage of each field contained within. Every object will have the unique identifier field marked; this is the field that should be chosen as the key identifier for the ECT when it is created in SharePoint Designer. Many of the fields may be read-only or nullable (an empty field when a value is not required); these fields will be marked as well.

9.1.1. Document Category

This object represents a document category within TechDoc. Categories are placed on documents in TechDoc to help define the sensitivity of the information contained within. Document categories are used to help decide things like the highest read access a document of this type can have, whether the document can be searchable and if so, what level of searchability it should have, etc. As the usage of this object in BCS is strictly as a supporting data type for the document object, it contains the just necessities:

- *documentCategoryAbbreviation* – **(Identifier) (Read-Only)** – This field stores the short hand abbreviation for a document category.
- *documentCategoryName* – **(Read-Only)** – The full name of the document category.

9.1.2. Document

This object represents a document in TechDoc. This object is by far the largest and most complex object in the TechDoc BCS service. This object contains all of the metadata of the document except the binary data. The binary can be accessed separately though the corresponding *Document_Fetch (get)* and *Document_CreateResidentDocument* or *Document_Replace (put)* operations. The metadata fields are as follows:

- *created* – **(Read-Only)** – The original creation date of the document in TechDoc.

- *documentCategoryAbbreviation* – The short hand abbreviation of the document category. This field should be connected to the document category ECT.

- *docID* – **(Identifier) (Read-Only)** – The unique ID based identifier for a document.

- *docKeywords* – **(Nullable)** – This is a special field that contains all of the keywords on a document and their corresponding values. This field is formatted using the following encoding:

  keywordName1=keyword1value|keyword2Name=keyword2Value

- *docNumber* – The user-friendly unique identifier for a document. This field is guaranteed to be unique, but is not the unique identifier that should be used for the ECT.

- *documentTypeAbbreviation* – The short hand abbreviation of the document type. This field should be connected to the document type ECT.

- *inReview* – **(Read-Only)** – This field is a Boolean value, yes or no that indicates whether or not a document is currently undergoing a review in TechDoc.

- *linkExploreDocument* – **(Read-Only)** – A hyperlink that can be accessed to show the "explore" view for a document in TechDoc.

- *linkFetchLatestGeneration* – **(Read-Only) (Nullable)** – A hyperlink that can be accessed to fetch (Download) the latest generation of the document from TechDoc.

- *linkFetchLatestRevision* – **(Read-Only) (Nullable)** – A hyperlink that can be accessed to fetch (Download) the latest released revision of a document from TechDoc.

- *modified* – **(Read-Only)** – The last modified date of the document in TechDoc.

- *nonresidentURL* – **(Nullable)** – The URL to the external resource for a nonresident document.

- *organizationAbbreviation* – The shorthand abbreviation of the organization this document belongs to in TechDoc.

- *owner* – **(Read-Only)** – The username of the owner of the document.

- *ownerDistribution* – A boolean value that denotes whether or not the owner should receive email notifications when a document is released for distribution.

- *ownerNotification* – A boolean value that denotes whether or not the owner should receive email notifications for general notifications on a document. Notifications are sent for actions on a document such as a document modification.

- *parentFolderPath* – **(Read-Only)** – The full path of the parent folder that this document resides in.

- *parentFolderID* – The unique identifier of the parent folder this this document resides in. This field should be mapped to the folder ECT.

- *pointOfContact* – **(Nullable)** – The point of contact for the document.

- *readAccess* – The read access setting for the document. This field should be mapped to the document read access ECT.

- *released* – **(Read-Only) (Nullable)** – The release date of the document if it has been released.

- *reservedBy* – **(Read-Only) (Nullable)** – This field will indicate the username of the individual the document is reserved by only when the document is reserved.

- *resident* – **(Read-Only)** – Tracks whether or not the document is resident meaning whether it has a file stored for it in TechDoc (resident) or not (nonresident).

- *revision* – **(Nullable)** – The latest revision of the document if it has been released.

- *title* – **(Nullable)** – The title of the document.

- *uuid* – **(Nullable)** – The uuid of the document. This field normally is not used in very many cases.

- *webSearch* – The web search value of the document. This field should be mapped to the document web search value ECT.

9.1.3. Document Keyword

This object represents the usage of a keyword on a document in TechDoc. Keywords are the most common thing individuals search on to locate documents in a TechDoc system. The metadata fields are as follows:

- *docID* – **(Read-Only)** – The unique identifier of the parent document this document keyword belongs to. This field should be mapped to the document ECT.

- *docKeywordID* – **(Identifier) (Read-Only)** – The unique identifier of this object.

- *docNumber* – The document number of the parent document. This field can be mapped to the document ECT.

- *keywordName* – The name of the parent keyword. This field can be mapped to the keyword ECT.

- *keywordValue* – The chosen value for this usage of the keyword.

9.1.4. Document Read Access

This is a simple object that is used only to return the legal read access values for use in a document read access field. This object contains just the single field:

- *readAccess* – **(Identifier) (Read-Only)** – The name of the this read access value.

9.1.5. Document Type

This object is used to represent a document type in TechDoc. Document types are used to separate documents by their type, decide whether or not their text should be indexed for searching and decide whether or not the render a document. Additionally, a records plan can be placed on a document type to automatically create a record when a document of that document type is created. A document type object contains the following fields:

- *documentTypeAbbreviation* – **(Identifier) (Read-Only)** – The short hand abbreviation for the document type.

- *documentTypeName* – **(Read-Only)** – The full name of the document type.

9.1.6. Document Web Search

This is a simple object that is used only to return the legal web search values to use for a document web search field. This object contains just the single field:

- *webSearch* – **(Identifier) (Read-Only)** – The name of the web search value.

9.1.7. Folder

This object represents a cabinet or folder in TechDoc. The folder object contains the following metadata fields:

- *created* – **(Read-Only)** – The original creation date of the folder in TechDoc.
- *description* – **(Nullable)** – The description of the folder.
- *folderID* – **(Identifier) (Read-Only)** – The unique identifier of the folder.
- *fullPath* – **(Read-Only)** – The fully qualified path of the folder in the TechDoc DM.
- *modified* – **(Read-Only)** – The last modified date of the folder in TechDoc.
- *name* – The name of the folder.
- *organizationAbbreviation* – The shorthand abbreviation of the organization the folder belongs to. This field should be mapped to the organization ECT.
- *owner* – **(Read-Only)** – The username of the TechDoc user that owns the folder.
- *ownerNotification* – A boolean value that denotes whether or not the owner should receive email notifications for general notifications on a folder. Notifications are sent for actions on a folder such as a folder modification.
- *parentPath* – The fully qualified path of the parent cabinet or folder the folder exists in.
- *readAccess* – The read access setting for the folder. This field should be mapped to the folder read access ECT.
- *rmaFilePlanName* – **(Nullable)** – The name of the RMA file plan on the folder if automatic records are being created for documents in the folder.
- *rmaRecordSetName* – **(Read-Only) (Nullable)** – The RMA records set this folder is a part of if any.
- *uuid* – **(Nullable)** – The uuid of the folder. This field normally is not used in very many cases.

9.1.8. Folder Read Access

This is a simple object that is used only to return the legal read access values for use in a folder read access field. This object contains just the single field:

- *readAccess* – **(Identifier) (Read-Only)** – The name of the this read access value.

9.1.9. Generation

This object represents a single generation of a document in TechDoc. All documents have at least one generation. This generation is representative of the physical file for that version of the document when the document is a resident. For non-resident documents, only a single generation may exist and no physical file is stored. Most of the time, for a non-resident document, the nonresident URL is used to point the location where the actual file is stored. The generation object contains the metadata fields below:

- *created* – **(Read-Only)** – The original creation date of the generation in TechDoc.
- *creator* – **(Read-Only)** – The username of the user that created the generation.
- *docNumber* – **(Read-Only)** – The document number of the parent document this generation belongs to. This field should be mapped to the document ECT.
- *encrypted* – **(Read-Only)** – A boolean value that denotes either true the file of the generation is stored encrypted while at rest, false otherwise.
- *fetchAccess* – Denote whether or not the generation has normal or restricted fetch access.
- *fileName* – **(Nullable)** – The name of the file for the generation if the generation is resident.
- *fileSize* – **(Nullable)** – The size in bytes of the file for the generation if the generation is resident.
- *genID* – **(Identifier) (Read-Only)** – The unique identifier of the generation.
- *genNumber* – **(Read-Only)** – The generation number of the generation.
- *mimeType* – **(Nullable)** – The mime type of the file of the generation if the generation is resident.

- *nonresidentURL* – **(Nullable)** – The nonresident URL is used only by nonresident generations to point to the location of the item the generation exists to represent.

- *released* – **(Read-Only) (Nullable)** – The release date of the generation if it has been released.

- *resident* – A boolean value that indicates either true, the generation is resident and has a local physical file or false, the generation is nonresident and does not have a local physical file.

- *revision* – **(Nullable)** – The revision given when the generation was released. Unreleased generations do not have a revision.

- *rmaRecord* – A boolean value that indicates either true, this generation has an associated RMA record to go along with it or false, this generation in not a part of any RMA record set.

9.1.10. Group

This is object represents a group in TechDoc. Groups are commonly used in TechDoc for association for access and other things in place of directly associating individual users to aid in user and access management. The group object contains the follow metadata fields:

- *created* – **(Read-Only)** – The original creation date of the group in TechDoc.

- *description* – **(Nullable)** – Describes the purpose of the group.

- *externalGroupID* – **(Nullable)** – The identifier string to use when mapping the TechDoc group to an external group. For example, if an external group is created in TechDoc to represent an Active Directory (AD) group, the AD SID would be entered in this field.

- *groupID* – **(Identifier) (Read-Only)** – The unique identifier of the group.

- *groupType* – Denotes whether the group is a private group (for use only by its creator) or a shared group that's for use by everyone in the system.

- *localMembers* – **(Nullable)** – A list of usernames of the local users that belong to the group.

- *memberCount* – **(Read-Only)** – The number of users in the group.

- *memberType* – Denotes the type of user the group represents: local users, remote users, or remote emails.
- *name* – The name of the group.
- *organizationAbbreviation* – The short hand abbreviation of the organization that owns the group. This field should be mapped to the organization ECT.
- *owner* – **(Read-Only)** – The username of the owner of the group.
- *remoteEmailMembers* – **(Nullable)** – The email addresses of the remote users that are identified by their email addresses.
- *remoteUserMembers* – **(Nullable)** – A list of entries of the remote users that are identified by their email addresses. Each entry in this list is separated by a vertical bar where each entry contains the authenticator and identifier separated by an equals sign.

9.1.11. **Keyword**

The keyword object represents a keyword in TechDoc. Keywords can be placed on a document when documents are created or modified to aid in searching for documents. A keyword represents the definition of the keyword itself where as a document keyword is the usage of a keyword on a document. The keyword object contains the following metadata fields:

- *datatype* – Denotes the type of data the keyword's value uses: string, date, number or URL.
- *description* – **(Nullable)** – A description of the keyword and its purpose.
- *inputType* – Denotes the type input the keyword is rendered with, either a dropdown box or free form input.
- *keywordID* – **(Identifier) (Read-Only)** – The unique identifier of the keyword.
- *name* – The name of the keyword.
- *private* – A boolean value that denotes whether or not the keyword is private.
- *values* – **(Nullable)** – The list of valid values the keyword can have when the input type is dropdown.

9.1.12. **Organization**

This object represents an organization in TechDoc. In TechDoc, users are typically organized into the different organizations or departments that they belong to. This object contains the following fields:

- *organizationAbbreviation* – **(Identifier) (Read-Only)** – The shorthand abbreviation of the organization.
- *organizationName* – **(Read-Only)** – The full name of the organization.

9.1.13. **RMA File Plan**

This object represents an RMA file plan in TechDoc. RMA file plans can be placed on folders and document types so that any time a document in created in the folder or of the document type, an automatic record is created for it. RMA file plan has the following metadata fields:

- *owner* – **(Read-Only)** – The username of the owner of the file plan.
- *permanent* – A boolean value that denotes whether or not the file plan is permanent.
- *recordCategoryDescription* – A description of the types of records the file plan is to contain.
- *recordCategoryIdentifier* – The internal identifier for the record category.
- *rmaFilePlanName* – **(Identifier) (Read-Only)** – The unique identifier for the RMA file plan.
- *vital* – A boolean value that denotes whether or not the file plan is vital.

9.1.14. **User**

This object represents a user in a TechDoc system that has an actual TechDoc user account. The user object contains many metadata fields:

- *emailAddress* – The email address of the user.
- *employerAbbrievation* – The shorthand abbreviation of the employer the user belongs to.
- *firstName* – The first name of the user.
- *lastName* – The last name of the user.
- *location* – **(Nullable)** – The location of the user.
- *mailCode* – **(Nullable)** – The inter-office mail code for the user.
- *middleInitial* – **(Nullable)** – The middle initial of the user.
- *organizationAbbreviation* – The shorthand abbreviation of the organization this user belongs to.
- *phoneNumber* – **(Nullable)** – The phone number of the user.
- *username* – **(Identifier) (Read-Only)** – The username of the user. Username is the identifier for a user throughout the TechDoc BCS service.
- *UUPIC* – **(Nullable)** – The UUPIC identifier for a user.

9.2. *BCS Operations*

This section will describe each of the web service operations available on the TechDoc BCS service. Each operation will detail the input parameters, which are required and which are optional, the return value if any, and so-on. All of the operations are sorted alphabetically first accordingly to their corresponding parent object type and then by the operation name.

All operations can throw a web service exception if invalid input parameters are specified or an expected error occurs. All of the read list type operations support the asterisk (*) wildcarding on all string based input parameters. Also, all of the operations that require custom code will be marked as non-standard. BCS requires all of the CRUD operations to give and take the same set of parameters. On certain operations such as create and update some parameters will be present as inputs to satisfy this requirement but will be ignored as they are not parameters that would normally be used for that corresponding TechDoc operation. Theses parameters will be marked as ignored.

BCS does not support an ECT having a child parameter that is an array. For instance, a document in TechDoc may have more than one keyword; this would not normally work in BCS. To get around this shortcoming, we'll accept and return a specially formatted string in places of these array based parameters. These encoded string arrays will/should be formatted as:

Keyword1=Keyword1Value|Keyword2=Keyword2Value

Where each key value pair is separated by a vertical bar and each key and value are separated by an equals sign. Any parameters that accept this type of value or any operations that return this type of value will be marked as an encoded string array.

9.2.1. DocumentCategory_ReadItemByAbbreviation

This read item operation is used to read an individual document category. This operation accepts the following input:

- *documentCategoryAbbreviation* – **(Required)** – The shorthand abbreviation of the document category.

This operation returns a single document category ECT instance.

9.2.2. DocumentCategory_ReadList

This read list operation is used to read a list of document categories. This operation accepts the following input:

- *documentCategoryAbbreviation* – The shorthand abbreviation of the document category.
- *documentCategoryName* – The name of the document category.

This operation returns a list of zero or more matching document category ECT instances.

9.2.3. DocumentKeyword_Create

This create operation is used to create a document keywords. The operation accepts the following input:

- *docID* – **(Ignored)** – The ID of the parent document.
- *docNumber* – **(Required)** – The document number of the parent document.
- *keywordName* – **(Required)** – The name of the keyword to use.
- *keywordValue* – **(Required)** – The value the document keyword should have.

This operation returns a complete document keyword ECT instance.

9.2.4. DocumentKeyword_Delete

This delete operation is used to delete a single document keyword. The operation accepts the following input:

- *dockeywordID* – **(Required)** – The ID of the document keyword to delete.

This operation has no return value.

9.2.5. DocumentKeyword_ReadItemById

This read item operation is used to read an individual document keyword. This operation accepts the following input:

- *docKeywordID* – **(Required)** – The ID of the document keyword.

This operation returns a single document keyword ECT instance.

9.2.6. DocumentKeyword_ReadList

This read list operation is used to read a list of document keywords. This operation accepts the following input:

- *keywordName* – The name of the parent keyword.
- *keywordValue* – The value of the document keyword.
- *docNumber* – The document number of the parent document.

This operation returns a list of zero or more matching document keyword ECT instances.

9.2.7. DocumentKeyword_Update

This update operation is used to update a single document keyword ECT instance. This operation accepts the following input:

- *docKeywordID* – **(Required)** – The ID of the document keyword.
- *docID* – **(Ignored)** – The ID of the parent document.
- *docNumber* – **(Required)** – The document number of the parent document.
- *keywordName* – **(Required)** – The name of the parent keyword.
- *keywordValue* – **(Required)** – The value of the document keyword.

9.2.8. DocumentReadAccess_ReadItemByName

This read item operation is used to read an individual document read access value. This operation accepts the following input:

- *readAccess* – **(Required)** – The name of the document read access value.

This operation returns a single document read access ECT instance.

9.2.9. DocumentReadAccess_ReadList

This read list operation is used to read a list of document read access values. This operation accepts the following input:

- *readAccess* – The name of the document read access value.

This operation returns a list of zero or more matching document read access ECT instances.

9.2.10. DocumentType_ReadItemByAbbreviation

This read item operation is used to read an individual document type. This operation accepts the following input:

- *documentTypeAbbreviation* – **(Required)** – The abbreviation of the document type.

This operation returns a single document type ECT instance.

9.2.11. DocumentType_ReadList

This read list operation is used to read a list of document types. This operation accepts the following input:

- *documentTypeAbbreviation* – The abbreviation of the document type.
- *documentTypeName* – The name of the document type.

This operation returns a list of zero or more matching document type ECT instances.

9.2.12. DocumentWebSearch_ReadItemByName

This read item operation is used to read an individual document web search value. This operation accepts the following input:

- *webSearch* – **(Required)** – The name of the document web search value.

This operation returns a single document web search ECT instance.

9.2.13. DocumentWebSearch_ReadList

This read list operation is used to read a list of document web search values. This operation accepts the following input:

- *webSearch* – The name of the document web search value.

This operation returns a list of zero or more matching document web search value ECT instances.

9.2.14. Document_AssociateAccess

This is a non-standard operation that can be used to associate the access on a document. This operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to associate access for.
- *groups* –An encoded string array of group names and the access values they should have or null if no groups should be associated.
- *users* – An encoded string array of usernames and the access values they should have or null if no users should be associated.
- *remoteUsers* – A normal string array of just remote users. This parameter does not accept access settings as remote users can only ever have read access. Null may be passed if no remote users should be associated for access.
- *readAccess* – **(Required)** – The read access setting the document should have. This could potentially be: None, Local, Campus, Community or Global. However, the document's document type can limit this, you'll need to add logic for this.

The access values should be specified as the value portion of the groups and/or users encoded string arrays using just one or two letters as follows:

- N – Denotes the access value of none.
- R – Denotes the access value of read.
- M – Denotes the access value of modify.
- D – Denotes the access value of delete.
- RR – Denotes the access value of reserve/replace.
- O – Denotes the access value of owner.

For example, to specify an encoded string array where a group named testGroup1 has an access setting of owner and testGroup2 has the access settings read, modify and reserve/replace, you would specify:

testGroup1=O|testGroup2=R,M,RR

In the example above you can see that multiple access values may be specified using a comma to separate them.

This operation has no return value.

9.2.15. **Document_AssociateCommenters**

This is a non-standard operation that can be used to associate the commenters on a document. This operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to associate commenters for.
- *groups* – An array of group names to associate for commenting on a document or null if no groups should be associated.
- *users* – An array of usernames to associate for commenting on a document or null if no users should be associated.

This operation has no return value.

9.2.16. **Document_AssociateDistribution**

This is a non-standard operation that can be used to associate the users for distribution on a document. This operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to associate distribution for.
- *groups* – An array of group names to associate for distribution on a document or null if no groups should be associated.
- *users* – An array of usernames to associate for distribution on a document or null if no users should be associated.
- *remoteEmails* – An array of email addresses to associate for distribution on a document or null if no remote emails should be associated.

This operation has no return value.

9.2.17. **Document_AssociateNotification**

This is a non-standard operation that can be used to associate the users for notification on a document. This operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to associate notification for.
- *groups* – An array of group names to associate for notification on a document or null if no groups should be associated.
- *users* – An array of usernames to associate for notification on a document or null if no users should be associated.
- *remoteEmails* – An array of email addresses to associate for notification on a document or null if no remote emails should be associated.

This operation has no return value.

9.2.18. **Document_Create**

This operation can be used to create a new nonresident document in TechDoc. This is the BCS friendly method of creating a document as it does not take the binary for the file. BCS does directly support working with SOAP standard binary and requires .Net shim code to deal with it. This operation accepts the following input:

- *created* – **(Ignored)**
- *documentCategoryAbbreviation* – **(Required)** – The abbreviation of the document category.
- *docKeywords* – An encoded string array of keywords and values.
- *docNumber* – **(Required)** – The document number for the newly created document.
- *documentTypeAbbreviation* – **(Required)** – The abbreviation of the document type.
- *inReview* – **(Ignored)**
- *modified* – **(Ignored)**
- *nonresidentURL* – This parameter can be used to specify a URL location for the physical file this document is to represent.
- *organizationAbbreviation* – **(Required)** – The abbreviation of the organization.
- *owner* – **(Ignored)**
- *ownerDistribution* – **(Required)** – A boolean value. True if the owner should be associated for distribution on the document, false otherwise.
- *ownerNotification* – **(Required)** – A boolean value. True if the owner should be associated for notification on the document, false otherwise.
- *parentFolderPath* – **(Required)** – The fully qualified path of the parent folder that this document should be created in.
- *parentFolderID* – **(Ignored)**
- *pointOfContact* – An optional field that can be used to specify a point of contact for the document.

- *readAccess* – **(Required)** – The document's read access setting. This can be: none, local, campus, community or global. However, the selected document type can lower the allowed access level.
- *released* – **(Ignored)**
- *reservedBy* – **(Ignored)**
- *resident* – **(Ignored)**
- *revision* – **(Ignored)**
- *title* – The title for the document.
- *uuid* – **(Ignored)**
- *webSearch* – **(Required)** – The web search value for the document. This can be: no, campus, community or global.

This operation returns a document ECT instance for the newly created document.

9.2.19. **Document_CreateResidentDocument**

This is a non-standard operation can be used to create a new resident document in TechDoc. This operation requires .Net shim code as this operation expects binary data. This operation accepts the following input:

- *created* – **(Ignored)**
- *documentCategoryAbbreviation* – **(Required)** – The abbreviation of the document category.
- *docKeywords* – An encoded string array of keywords and values.
- *docNumber* – **(Required)** – The document number for the newly created document.
- *documentTypeAbbreviation* – **(Required)** – The abbreviation of the document type.
- *inReview* – **(Ignored)**
- *modified* – **(Ignored)**
- *nonresidentURL* – This parameter can be used to specify a URL location for the physical file this document is to represent.
- *organizationAbbreviation* – **(Required)** – The abbreviation of the organization.
- *owner* – **(Ignored)**
- *ownerDistribution* – **(Required)** – A boolean value. True if the owner should be associated for distribution on the document, false otherwise.
- *ownerNotification* – **(Required)** – A boolean value. True if the owner should be associated for notification on the document, false otherwise.
- *parentFolderPath* – **(Required)** – The fully qualified path of the parent folder that this document should be created in.
- *parentFolderID* – **(Ignored)**
- *pointOfContact* – An optional field that can be used to specify a point of contact for the document.

- *readAccess* – **(Required)** – The document's read access setting. This can be: none, local, campus, community or global. However, the selected document type can lower the allowed access level.

- *released* – **(Ignored)**

- *reservedBy* – **(Ignored)**

- *resident* – **(Ignored)**

- *revision* – **(Ignored)**

- *title* – The title for the document.

- *uuid* – **(Ignored)**

- *webSearch* – **(Required)** – The web search value for the document. This can be: no, campus, community or global.

- *fetchAccess* – The fetch access setting for the document. This can be either normal or restricted. This field is required only when creating a resident document.

- *FILE_DATA_STREAM* – **(Required)** – This parameter can have different names depending on the client-side consumption of the web service code. This parameter should contain the binary data stream for the file or null if a nonresident document is being created.

- *generationFileName* – The filename to use for the generation. This is the filename that will be given to the physical file when a user attempts to fetch (download) the document. Again, this can be null only when using this operation to create a nonresident document.

- *mimeType* – The mime type to use for this file. This parameter can be left null to allow TechDoc to attempt to determine the mime type on its own.

This operation returns a document ECT instance for the newly created document.

9.2.20. **Document_Delete**

This delete operation is used to delete a single document. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to delete.

This operation has no return value.

9.2.21. **Document_DeleteComment**

This is a non-standard operation that can be used to delete a single document comment. The operation accepts the following input:

- *docCommentID* – **(Required)** – The ID of the document comment to delete.

This operation has no return value.

9.2.22. **Document_ExistsByDocNumber**

This is a non-standard convenience operation that can be used to check if a single document exists in TechDoc using its document number. The operation accepts the following input:

- *docNumber* – **(Required)** – The document number of the document to check.

This operation returns true if the document exists, false otherwise.

9.2.23. **Document_ExistsByID**

This is a non-standard convenience operation that can be used to check if a single document exists in TechDoc using its ID. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to check.

This operation returns true if the document exists, false otherwise.

9.2.24. Document_Fetch

This is a non-standard operation that can be used to fetch the binary file for a resident document. BCS doesn't support standard SOAP binary directly and as such custom .Net shim code is needed. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to fetch.
- *genID* – **(Required)** – The ID of the generation to fetch.

This operation returns a SOAP standard binary data stream.

9.2.25. Document_GetGenerations

This is a non-standard convenience operation that can be used to list all of the generations of a document. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to check.

This operation returns an array of generation IDs.

9.2.26. Document_ReadItemById

This read item operation is used to read an individual document. This operation accepts the following input:

- *docID* – **(Required)** – The ID of the document.

This operation returns a single document ECT instance.

9.2.27. **Document_ReadList**

This read list operation is used to read a list of documents. This operation accepts the following input:

- *comments* – A string value indicating if the document has any comments. This can be: no comments, closed comments or open comments.
- *created* – The create date of the document.
- *documentCategoryAbbreviation* – The document category abbreviation.
- *docNumber* – The document number of the document.
- *docTypeAbbreviation* – The document type abbreviation.
- *lastestRevision* – The latest revision of the document.
- *modified* – The last modified date of the document.
- *organizationAbbreviation* – The organization abbreviation.
- *owner* – The username of the owner.
- *parentFolderPath* – The fully qualified path of the parent folder the document exists in.
- *pointOfContact* – The point of contact for the document.
- *readAccess* – The read access on the document.
- *released* – The release date of the document.
- *reservedBy* – The username of the user that has the document reserved.
- *title* – The title of the document.
- *webSearch* – The web search setting on the document.

This operation returns a list of zero or more matching document ECT instances.

9.2.28. **Document_Release**

This is a non-standard operation that can be used to release a document in TechDoc. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document.
- *genID* – **(Required)** – The ID of the generation that should be released.
- *revision* – **(Required)** – The released revision to set on the generation.
- *render* – **(Required)** – True if the document should attempt to render to a PDF, false otherwise. Note: not all doc types/file types support this. Only those allowed and supported will attempt to render to a PDF.
- *metricStatus* – The metric status to associate on the released generation. This only applies to documents with a metric doc type.
- *metricDate* – The metric date to place on the released generation. This only applies to documents with a metric doc type.

This operation has no return value.

9.2.29. **Document_Replace**

This is a non-standard operation that can be used to replace a document in TechDoc. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document.
- *keepReserved* – **(Required)** – True if the document should remain reserved after being replaced, false to let it unreserved.
- *overwrite* – **(Required)** – True if the current generation should be overwritten with the new file, false if an additional newer generation should be created for the file.
- *fetchAccess* – **(Required)** – The fetch access to place on the generation either normal or restricted.
- *genFileName* – **(Required)** – The filename for the generation.
- *FILE_DATA_STREAM* – **(Required)** – This parameter can have different names depending on the client-side consumption of the web service code. This parameter should contain the binary data stream for the file.

- *mimeType* – **(Required)** – The mime type of the file or null to let TechDoc try and determine the correct mime type using the file extension.

This operation has no return value.

9.2.30. Document_Reserve

This is a non-standard operation that can be used to reserve a single document in TechDoc. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document.

This operation has no return value.

9.2.31. Document_ShowAssociations

This is a non-standard operation that can be used to view the associations on a document. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document.

This operation returns a complex *DocumentAssocation* object that contains all of the information for the various types of associations that may be placed on a document.

9.2.32. Document_Unrelease

This is a non-standard operation that can be used to unrelease a single document in TechDoc. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document.
- *genID* – **(Required)** – The ID of the generation to unrelease.

This operation has no return value.

9.2.33. **Document_Unreserve**

This is a non-standard operation that can be used to unreserve a single document in TechDoc. The operation accepts the following input:

- *docID* – **(Required)** – The ID of the document.

This operation has no return value.

9.2.34. **Document_Update**

This operation can be used to update a document in TechDoc. This operation accepts the following input:

- *docID* – **(Required)** – The ID of the document to update.
- *created* – **(Ignored)**
- *documentCategoryAbbreviation* – **(Required)** – The abbreviation of the document category.
- *docKeywords* – An encoded string array of keywords and values.
- *docNumber* – **(Required)** – The document number for the newly created document.
- *documentTypeAbbreviation* – **(Required)** – The abbreviation of the document type.
- *inReview* – **(Ignored)**
- *modified* – **(Ignored)**
- *nonresidentURL* – This parameter can be used to specify a URL location for the physical file this document is to represent.
- *organizationAbbreviation* – **(Required)** – The abbreviation of the organization.
- *owner* – **(Ignored)**
- *ownerDistribution* – **(Required)** – A boolean value. True if the owner should be associated for distribution on the document, false otherwise.
- *ownerNotification* – **(Required)** – A boolean value. True if the owner should be associated for notification on the document, false otherwise.

- *parentFolderPath* – **(Required)** – The fully qualified path of the parent folder that this document should be created in.
- *parentFolderID* – **(Ignored)**
- *pointOfContact* – An optional field that can be used to specify a point of contact for the document.
- *readAccess* – **(Required)** – The document's read access setting. This can be: none, local, campus, community or global. However, the selected document type can lower the allowed access level.
- *released* – **(Ignored)**
- *reservedBy* – **(Ignored)**
- *resident* – **(Ignored)**
- *revision* – **(Ignored)**
- *title* – The title for the document.
- *uuid* – **(Ignored)**
- *webSearch* – **(Required)** – The web search value for the document. This can be: no, campus, community or global.

This operation has no return value.

9.2.35. FolderReadAccess_ReadItemByName

This read item operation is used to read an individual folder read access value. This operation accepts the following input:

- *readAccess* – **(Required)** – The name of the folder read access value.

This operation returns a single folder read access ECT instance.

9.2.36. **FolderReadAccess_ReadList**

This read list operation is used to read a list of folder read access values. This operation accepts the following input:

- *readAccess* – The name of the folder read access value.

This operation returns a list of zero or more matching folder read access ECT instances.

9.2.37. **Folder_AssociateAccess**

This is a non-standard operation that can be used to associate the access on a folder. This operation accepts the following input:

- *folderID* – **(Required)** – The ID of the folder to associate access for.
- *groups* –An encoded string array of group names and the access values they should have or null if no groups should be associated.
- *users* – An encoded string array of usernames and the access values they should have or null if no users should be associated.
- *readAccess* – **(Required)** – The read access setting the folder should have. This could be none or local.
- *folderInheritanceType* – **(Required)** – The folder inheritance type to set on the folder. Nothing, replace, merge or override.
- *documentInheritanceType* – **(Required)** – The document inheritance type to set on the folder. Nothing, replace, merge or override.

The access values should be specified as the value portion of the groups and/or users encoded string arrays using just one or two letters as follows:

- N – Denotes the access value of none.
- R – Denotes the access value of read.
- M – Denotes the access value of modify.
- D – Denotes the access value of delete.
- CF – Denotes the access value of create folder.

- CD – Denotes the access value of create document.
- O – Denotes the access value of owner.

For example, to specify an encoded string array where a group named testGroup1 has an access setting of owner and testGroup2 has the access settings read and modify, you would specify:

testGroup1=O|testGroup2=R,M

In the example above you can see that multiple access values may be specified using a comma to separate them.

This operation has no return value.

9.2.38. **Folder_Create**

This create operation is used to create a folder. The operation accepts the following input:

- *created* – **(Ignored)**
- *description* – A description for the folder.
- *fullPath* – **(Ignored)**
- *modified* – **(Ignored)**
- *name* – **(Required)** – The name for the folder.
- *organizationAbbreviation* – **(Required)** – The organization abbreviation.
- *owner* – **(Ignored)**
- *ownerNotification* – **(Required)** – True if the owner should be associated for notification on the folder, false otherwise.
- *parentPath* – **(Required)** – The fully qualified path of the parent cabinet or folder that the folder should be created in.
- *readAccess* – **(Required)** – The read access for the folder either none or local.
- *rmaFilePlanName* –The name of the RMA file plan to place on the folder.
- *rmaRecordSetName* – **(Ignored)**
- *uuid* – **(Ignored)**

This operation returns a complete folder ECT instance.

9.2.39. **Folder_CreateFolderTree**

This non-standard create operation is used to create a folder. It works the same as the standard *Folder_Create* operation exception that if intermediate folders are missing in the parent path specified, they will be created before the target folder is created. In this way, a verify doesn't need to be perform on each piece (folder) of the path before creating the target folder. The standard *Folder_Create* operation will fail if any parts of the parent path do not already exist. The operation accepts the following input:

- *created* – **(Ignored)**
- *description* – A description for the folder.
- *fullPath* – **(Ignored)**
- *modified* – **(Ignored)**
- *name* – **(Required)** – The name for the folder.
- *organizationAbbreviation* – **(Required)** – The organization abbreviation.
- *owner* – **(Ignored)**
- *ownerNotification* – **(Required)** – True if the owner should be associated for notification on the folder, false otherwise.
- *parentPath* – **(Required)** – The fully qualified path of the parent cabinet or folder that the folder should be created in.
- *readAccess* – **(Required)** – The read access for the folder either none or local.
- *rmaFilePlanName* –The name of the RMA file plan to place on the folder.
- *rmaRecordSetName* – **(Ignored)**
- *uuid* – **(Ignored)**

This operation returns a complete folder ECT instance.

9.2.40. **Folder_Delete**

This delete operation is used to delete a single folder. The operation accepts the following input:

- *folderID* – **(Required)** – The ID of the folder to delete.

This operation has no return value.

9.2.41. **Folder_ExistsByID**

This is a non-standard convenience operation that can be used to check if a single folder exists in TechDoc using its ID. The operation accepts the following input:

- *folderID* – **(Required)** – The folder ID of the folder to check.

This operation returns true if the folder exists, false otherwise.

9.2.42. **Folder_ExistsByPath**

This is a non-standard convenience operation that can be used to check if a single folder exists in TechDoc using its full path. The operation accepts the following input:

- *folderPath* – **(Required)** – The fully qualified path of the folder to check.

This operation returns true if the folder exists, false otherwise.

9.2.43. **Folder_GetChildDocuments**

This is a non-standard convenience operation that can be used to get a list of the child documents contained in a folder in TechDoc. The operation accepts the following input:

- *folderID* – **(Required)** – The ID of the folder to check.

This operation returns an array of zero or more document IDs.

9.2.44. **Folder_GetChildFolders**

This is a non-standard convenience operation that can be used to get a list of the child folders contained in a folder in TechDoc. The operation accepts the following input:

- *folderID* – **(Required)** – The ID of the folder to check.

This operation returns an array of zero or more folder IDs.

9.2.45. **Folder_MoveContents**

This is a non-standard operation that can be used to move documents and folder inside a cabinet or folder to another cabinet or folder in TechDoc. The operation accepts the following input:

- *srcFolderID* – **(Required)** – The ID of the parent folder where the items will be sourced from.
- *destFolderID* – **(Required)** – The ID of the folder that should receive the child folders and documents.
- *foldersToMove* – **(Required)** – An array of folder IDs to move or null if no child folders are to be moved.
- *documentsToMove* – **(Required)** – An array of document IDs to move or null if no child documents are to be moved.

This operation returns the total count of folders and documents moved.

9.2.46. **Folder_ReadItemById**

This read item operation is used to read an individual folder. This operation accepts the following input:

- *folderID* – **(Required)** – The ID of the folder.

This operation returns a single folder ECT instance.

9.2.47. **Folder_ReadList**

This read list operation is used to read a list of folders. This operation accepts the following input:

- *description*– The description of the folder.
- *fullPath* – The full path of the folder.
- *name* – The folder name.
- *organizationAbbreviation* – The shorthand abbreviation of the organization.
- *owner* – The username of the owner of the folder.
- *parentPath* – The full path of the parent cabinet or folder that contains this folder.

This operation returns a list of zero or more matching folder ECT instances.

9.2.48. **Folder_Update**

This modify operation is used to update a folder. The operation accepts the following input:

- *folderID* – **(Required)** – The ID of the folder to update.
- *created* – **(Ignored)**
- *description* – A description for the folder.
- *fullPath* – **(Ignored)**
- *modified* – **(Ignored)**
- *name* – **(Required)** – The name for the folder.
- *organizationAbbreviation* – **(Required)** – The organization abbreviation.
- *owner* – **(Ignored)**
- *ownerNotification* – **(Required)** – True if the owner should be associated for notification on the folder, false otherwise.

- *parentPath* – **(Required)** – The fully qualified path of the parent cabinet or folder that the folder should be created in.
- *readAccess* – **(Required)** – The read access for the folder either none or local.
- *rmaFilePlanName* –The name of the RMA file plan to place on the folder.
- *rmaRecordSetName* – **(Ignored)**
- *uuid* – **(Ignored)**

This operation has no return value.

9.2.49. Generation_CreateRendition

This is a non-standard operation that can be used to manually create a rendition for a generation in TechDoc. The operation accepts the following input:

- *docID*– **(Required)** – The ID of the parent document.
- *genID* – **(Required)** – The ID of the generation.
- *FILE_DATA_STREAM* – **(Required)** – This parameter can have different names depending on the client-side consumption of the web service code. This parameter should contain the binary data stream for the file.
- *genFileName* – **(Required)** – The file name of the file being uploaded.
- *fetchAccess* – **(Required)** – The fetch access to set on the generation either normal or restricted.

This operation returns the total count of folders and documents moved.

9.2.50. Generation_Delete

This delete operation is used to delete a single generation. The operation accepts the following input:

- *genID* – **(Required)** – The ID of the generation to delete.

This operation has no return value.

9.2.51. Generation_ExistsByGenNumber

This is a non-standard convenience operation that can be used to check if a single generation exists in TechDoc using its generation number. The operation accepts the following input:

- *docNumber* – **(Required)** – The document number of the parent document.
- *genNumber* – **(Required)** – The generation number of the generation number to look for.
- *genType* – **(Required)** – The type of generation to check for, either native or rendered.

This operation returns true if the generation exists, false otherwise.

9.2.52. Generation_ExistsByID

This is a non-standard convenience operation that can be used to check if a single generation exists in TechDoc using its ID. The operation accepts the following input:

- *genID* – **(Required)** – The ID of the generation to check.

This operation returns true if the generation exists, false otherwise.

9.2.53. Generation_ReadItemById

This read item operation is used to read an individual generation. This operation accepts the following input:

- *genID* – **(Required)** – The ID of the generation.

This operation returns a single generation ECT instance.

9.2.54. Generation_ReadList

This read list operation is used to read a list of generations. This operation accepts the following input:

- *docNumber* – The parent document number.

This operation returns a list of zero or more matching generation ECT instances.

9.2.55. Generation_ReplaceRendition

This is a non-standard convenience operation that can be used to manually replace a rendition for a generation in TechDoc. The operation accepts the following input:

- *genID* – **(Required)** – The ID of the generation to replace.
- *FILE_DATA_STREAM* – **(Required)** – This parameter can have different names depending on the client-side consumption of the web service code. This parameter should contain the binary data stream for the file.
- *genFileName* – **(Required)** – The file name of the file being uploaded.
- *fetchAccess* – **(Required)** – The fetch access to set on the generation either normal or restricted.

This operation has no return value.

9.2.56. Generation_ResubmitRendition

This is a non-standard convenience operation that can be used to manually resubmit a generation for rendition in TechDoc. The operation accepts the following input:

- *genID* – **(Required)** – The ID of the generation to replace.

This operation has no return value.

9.2.57. **Generation_Update**

This is a standard update operation that can be used to update a generation in TechDoc however an ECT cannot be created for a generation in BCS as generation cannot be created directly in TechDoc; a create document operation must be used instead. The operation accepts the following input:

- *genID* – **(Required)** – The ID of the generation to update.
- *created* – **(Ignored)**
- *creator* – **(Ignored)**
- *docNumber* – **(Ignored)**
- *encrypted* – **(Ignored)**
- *fetchAccess* – **(Required)** – The fetch access to set on the generation, either normal or restricted.
- *fileName* – **(Required)** – The file name to set on the file for the generation. This is the name that will be given when the generation is downloaded.
- *fileSize* – **(Ignored)**
- *genNumber* – **(Ignored)**
- *mimeType* – **(Required)** – The mime type of the file or null to let TechDoc try and determine the mime type by the extension of the filename.
- *nonresidentURL* – The nonresident URL for the generation if it is a nonresident generation.
- *released* – **(Ignored)**
- *resident* – **(Ignored)**
- *revision* – **(Required)** – The revision of the generation.
- *rmaRecord* – **(Ignored)**

This operation has no return value.

9.2.58. Generation_UpdateReleaseDate

This is a non-standard operation that can be used to update released date of a released generation in TechDoc. The operation accepts the following input:

- *genID* – **(Required)** – The ID of the generation.
- *releaseDate* – **(Required)** – The new release date.

This operation has no return value.

9.2.59. Group_ReadItemByName

This read item operation is used to read an individual group. This operation accepts the following input:

- *groupName* – **(Required)** – The name of the group.

This operation returns a single group ECT instance.

9.2.60. Group_ReadList

This read list operation is used to read a list of groups. This operation accepts the following input:

- *groupName* – The name of the group.

This operation returns a list of zero or more matching group ECT instances.

9.2.61. Keyword_ReadItemByName

This read item operation is used to read an individual keyword. This operation accepts the following input:

- *keywordName* – **(Required)** – The name of the keyword.

This operation returns a single keyword ECT instance.

9.2.62. **Keyword_ReadList**

This read list operation is used to read a list of keywords. This operation accepts the following input:

- *keywordName* – The name of the keyword.

This operation returns a list of zero or more matching keyword ECT instances.

9.2.63. **Organization_ReadItemByAbbreviation**

This read item operation is used to read an individual organization. This operation accepts the following input:

- *organizationAbbreviation* – **(Required)** – The shorthand abbreviation of the organization.

This operation returns a single organization ECT instance.

9.2.64. **Organization_ReadList**

This read list operation is used to read a list of organization. This operation accepts the following input:

- *organizationAbbreviation* – The shorthand abbreviation of the organization.
- *organizationName* – The full name of the organization.

This operation returns a list of zero or more matching organization ECT instances.

9.2.65. **RmaFilePlan_ReadItemByName**

This read item operation is used to read an individual RMA file plan. This operation accepts the following input:

- *rmaFilePlanName* – **(Required)** – The full name of the RMA file plan.

This operation returns a single RMA file plan ECT instance.

9.2.66. **RmaFilePlan_ReadList**

This read list operation is used to read a list of RMA file plans. This operation accepts the following input:

- *rmaFilePlanName* – The full name of the RMA file plan.

This operation returns a list of zero or more matching RMA file plan ECT instances.

9.2.67. **User_ReadItemByUsername**

This read item operation is used to read an individual user. This operation accepts the following input:

- *username* – **(Required)** – The username of the user.

This operation returns a single user ECT instance.

9.2.68. **User_ReadList**

This read list operation is used to read a list of users. This operation accepts the following input:

- *username* – The username of the user.

This operation returns a list of zero or more matching user ECT instances.

10. Suggestions and Feedback

We hope you find the workflow engine to be very useful in automating processes in TechDoc. As the workflow engine is quite vast and very new, we have chosen to expose just the TechDoc operations outlined in this guide for this version. As we continue to build upon the workflow engine, we can begin to add more operations. If you have any suggestions about operations that should be added, how the workflow engine sections are laid out in TechDoc, etc.., please let us know by sending us email at suggestions@docubrain.com

If you do happen to find a problem, please feel free to use the same email above. While we do prefer suggestions and compliments, we also accept criticism (preferably constructive :-)

At Prevo Technologies, Inc., we take the quality of our products very seriously. If you do find an issue or something that you feel can be improved upon in the TechDoc workflow engine, please let us know.