HITEC LABORATORIES

DataStore®DSX Document Handling API Developers' Guide

APPROVAL AND PUBLISHING HISTORY

Version 1.0.0.0	Initial Release. Approved and Published.	3rd February 2014
Version 1.0.0.1	Minor changes made. Approved and Published.	5th February 2014
Version 1.0.0.2	The API can now handle versioned document identifiers.	11th April 2014
Version 1.0.0.3	Section 4, COM-based API information added.	30th April 2014
Version 1.0.0.4	Changes made to COM API after testing revealed some issues.	9th May 2014
Version 1.1.0.0	Changes made to all APIs, comprising fixes and additions.	2nd October 2014
Version 3.0.0.0	Support for versioned identifiers. Added accuracy information on date and time.	12th February 2015
Version 3.0.0.1	DSX API now throws specific exception whenever a document does not exist.	9th May 2016
Version 3.0.0.2	DSX API now returns MIME Type when Document Properties are requested	27th July 2016

Hitec (Laboratories) Ltd welcomes customer comments as part of the process of continuous development and improvement of its documentation.

To contact us, email: technical.documentation@hiteclabs.com

Contents

1 Introduction

This document provides a technical overview of how the Document Handling Service API can be used to integrate with the DataStoreDSX document management system.

The DataStoreDSX Document Handling API provides a selection of methods by which a third party can inject, search for, retrieve and update documents in DataStoreDSX.

The API consists of the following interfaces:

- A web service that exposes a SOAP-compatible web interface.
- A COM-compatible programmatic interface.
- A C#.NET programmatic interface.

This document assumes the reader is familiar with DataStoreDSX.

2 General Information

The following information applies to all of the APIs.

2.1 Date/Time Accuracy

It is possible to set up a data definition inside DataStoreDSX with a date/time field of configurable accuracy. For example, a "date of birth" field can be defined with day accuracy.

Due to the limitations on date/time data types available in the various interfaces, the actual values returned are full date/time values. For example, 15th April 2014 may come back as 15/04/2014 01:00:00. Consumers of the APIs therefore must know what accuracy is required.

3 Document Handling Web Service API

The Document Handling web service provides third parties with direct access to a DataStoreDSX system.

This service provides the ability to use DataStoreDSX to store, retrieve, search, update and delete data items.

All methods inside the API are synchronous, and will block until completion.

Should an error occur, detailed error information is returned (as faults/exceptions) by the web service.

3.1 Web Service API - Complex Data Types

The document handling web service API makes use of the following complex data types.

3.1.1 Credentials

Each method takes in a *Credentials* object which has two known types: *CASCredentials* and *UserNameCredentials*. If null is passed in, the service will assume that Windows Authentication is being used.

3.1.1.1 CASCredentials

Member	Data Type	Mandatory	Comments
ServiceTicket	string	Yes	The CAS service ticket
TicketGrantingTicket	string	Yes	The CAS ticket granting ticket

3.1.1.2 UserNameCredentials

Member	Data Type	Mandatory	Comments
UserName	string	Yes	The user name of the user
Password	string	Yes	The password of the user

3.1.2 MetadataItem

The *MetadataItem* can be used to pass metadata into DataStoreDSX for searching, storing and updating purposes. It is also used to retrieve metadata from DataStoreDSX. A *MetadataItem* consists of a field name and value. The field name is the name given to the custom field type, which must be provided for each *MetadataItem*. The value can be of type *string*, *DateTime*, *integer*, or *decimal* (as appropriate to the Custom Field Type).

Member	Data Type	Mandatory	Comments
FieldName	string	Yes	The name of the field in DataStoreDSX
Value	object[]	No	The list of values of the metadata field

3.1.3 DocumentDoesNotExist

Most methods involve getting and using a document from DataStoreDSX. If the document does not exist, an exception is thrown. To easily distinguish these types of exceptions from others the *DocumentDoesNotExist* object can be used as a *FaultType* in the returned *FaultException*.

Member	Data Type	Mandatory	Comments
Message	string	Yes	The original exception message.

```
catch (FaultException<DocumentDoesNotExist> docException)
{
   // Perform specific handling for documents not found
}
Catch (Exception ex)
{
   // Perform specific handling for any other exceptions
}
```

3.2 Storing a New Document

The following available methods are associated with storing new document data:

- StoreNewRevisionedDocument
- StoreNewRevisionedDocumentSets
- StoreNewVersionedDocument
- StoreNewVersionedDocumentSets
- StoreNewVersionedDocumentIdentifier
- StoreNewVersionedDocumentSetsIdentifier
- StoreNewVersionedDocumentVersion
- StoreNewVersionedDocumentSetsVersion

3.2.1 StoreNewRevisionedDocument[Sets]

The methods to store a new document and its associated metadata, returning a document revision identifier, have the following signatures:

```
void StoreNewRevisionedDocument (Credentials credentials,
                                 string dataDefinitionName,
                                 byte[] document,
                                 Metadataitem[] metadata,
                                 out string revisionID);
```

```
void StoreNewRevisionedDocumentSets (Credentials credentials,
                                     string dataDefinitionName,
                                     byte[] document,
                                     Metadataitem[][] metadata,
                                     out string revisionID);
```

credentials	The credentials to verify the calling user for this call.
dataDefinitionName	The name of the Data Definition which is used to store the new document.
document	The document contents to store in DataStoreDSX.
metadata	The metadata associated with this document. Method names without the word "Sets" in them accept a single dimension array representing a single index set, while method names with the word "Sets" in them accept a two dimensional array representing multiple index sets.
out revisionID	The unique revision identifier assigned by DataStoreDSX when the document is stored.
return void	Nothing.

3.2.2 StoreNewVersionedDocument[Sets][Identifier|Version]

The methods to store a new specifically versioned document and its associated metadata have the following signatures:

```
void StoreNewVersionedDocument (Credentials credentials,
                                string dataDefinitionName,
                                byte[] document,
                                Metadataitem[] metadata,
                                out Guid documentID,
                                out string documentVersion);
```

```
void StoreNewVersionedDocumentSets (Credentials credentials,
                                    string dataDefinitionName,
                                    byte[] document,
                                    Metadataitem[][] metadata,
                                    out Guid documentID,
                                    out string documentVersion);
```

```
void StoreNewVersionedDocumentIdentifier (Credentials credentials,
                                          string dataDefinitionName,
                                          byte[] document,
                                          Metadataitem[] metadata,
                                          Guid documentID,
                                          out string documentVersion);
```

```
void StoreNewVersionedDocumentSetsIdentifier (Credentials credentials,
                                              string dataDefinitionName,
                                              byte[] document,
                                              Metadataitem[][] metadata,
                                              Guid documentID,
                                              out string documentVersion);
```

```
void StoreNewVersionedDocumentVersion (Credentials credentials,
                                       string dataDefinitionName,
```

```
                                        byte[] document,
                                        Metadataitem[] metadata,
                                        Guid documentID,
                                        string documentVersion);
```

```
void StoreNewVersionedDocumentSetsVersion (Credentials credentials,
                                           string dataDefinitionName,
                                           byte[] document,
                                           Metadataitem[][] metadata,
                                           Guid documentID,
                                           string documentVersion);
```

credentials	The credentials to verify the calling user for this call.
dataDefinitionName	The name of the Data Definition which is used to store the new document.
document	The document contents to store in DataStoreDSX.
metadata	The metadata associated with this document. Method names without the word "Sets" in them accept a single dimension array representing a single index set, while method names with the word "Sets" in them accept a two dimensional array representing multiple index sets.
out documentID or documentID	The unique document identifier. Depending on the method being called, it is either assigned by the caller or assigned by DataStoreDSX.
out documentVersion or documentVersion	The version number of the document. Depending on the method being called, it is either assigned by the caller or assigned by DataStoreDSX.
return void	Nothing.

3.3 Retrieving Document Data

The following available methods are associated with the retrieval of document data:

- GetDocumentRevision
- GetDocumentVersion
- GetDocumentMetadataRevision
- GetDocumentMetadataSetsRevision
- GetDocumentMetadataVersion
- GetDocumentMetadataSetsVersion
- GetDocumentPropertiesRevision
- GetDocumentPropertiesVersion
- GetDocumentVersionHistory

3.3.1 GetDocument{Revision|Version}

The method to retrieve the contents of the document based on revision identifier has the following signature:

```
void GetDocumentRevision (Credentials credentials,
                          string revisionID,
                          string mimeType,
                          int[] pages,
                          out byte[] data);
```

The method to retrieve the contents of the document based on document identifier and version number has the following signature:

```
void GetDocumentVersion (Credentials credentials,
                         Guid documentID,
                         string documentVersion,
                         string mimeType,
                         int[] pages,
                         out byte[] data);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to be retrieved from DataStoreDSX.
documentID	The unique document identifier of the document to be retrieved from DataStoreDSX.
documentVersion	The version number of the document.
mimeType	The mime type to return the document as. Pass as null to return the document in its stored format.
pages	The individual pages to retrieve from the document (if applicable e.g. 1, 3, 4 returns pages 1, 3 and 4). Pass as null to return all pages.
out data	The contents of the document will be available in this parameter after retrieval is complete.
return void	Nothing.

3.3.2 GetDocumentMetadata[Sets]{Revision|Version}

The methods to retrieve document metadata based on a revision identifier have the following signatures:

```
void GetDocumentMetadataRevision (Credentials credentials,
                                  string revisionID,
                                  out MetadataItem[] metadata);
```

```
void GetDocumentMetadataSetsRevision (Credentials credentials,
                                      string revisionID,
                                      out MetadataItem[][] metadata);
```

The methods to retrieve document metadata from a specific version of a document have the following signatures:

```
void GetDocumentMetadataVersion (Credentials credentials,
                                 Guid documentID,
                                 string documentVersion,
                                 out MetadataItem[] metadata);
```

```
void GetDocumentMetadataSetsVersion (Credentials credentials,
                                     Guid documentID,
                                     string documentVersion,
                                     out MetadataItem[][] metadata);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document for which to retrieve metadata.
documentID	The unique document identifier of the document for which to retrieve metadata.
documentVersion	The version number of the document.
out metadata	The metadata of the document will be available in this parameter after retrieval is complete. Method names *without* the word "Sets" in them return a single dimension array representing the document's first index set, while method names *with* the word "Sets" in them return a two dimensional array containing all of the document's index sets.
return void	Nothing.

3.3.3 GetDocumentProperties{Revision|Version}

The methods to retrieve document properties based on a revision identifier have the following signatures:

```
void GetDocumentPropertiesRevision (Credentials credentials,
                                    string revisionID,
                                    out MetadataItem[] metadata);
```

```
void GetDocumentPropertiesRevision (Credentials credentials,
                                    string revisionID,
                                    out MetadataItem[] metadata,
                                    out string mimeType);
```

The methods to retrieve document properties based on the version of a document have the following signatures:

```
void GetDocumentPropertiesVersion (Credentials credentials,
                                   Guid documentID,
                                   string documentVersion,
                                   out MetadataItem[] metadata);
```

```
void GetDocumentPropertiesVersion (Credentials credentials,
                                   Guid documentID,
                                   string documentVersion,
                                   out MetadataItem[] metadata,
                                   out string mimeType);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document for which to retrieve properties.
documentID	The unique document identifier of the document for which to retrieve properties.
documentVersion	The version number of the document.
out metadata	The properties of the document will be available in this parameter after retrieval is complete.
out mimeType	The format of the specified document.
return void	Nothing.

3.3.4 GetDocumentVersionHistory

The method to retrieve version numbers associated with a document has the following signature:

```
string[] GetDocumentVersionHistory (Credentials credentials,
                                    Guid documentID);
```

credentials	The credentials to verify the calling user for this call.
documentID	The unique document identifier of the document whose version numbers to retrieve.
return string[]	Returns the list of version numbers associated with the specified document.

3.4 Searching Document Data

The following available methods are associated with the searching of document data:

- Search
- GetSearchResultsRevision
- GetSearchResultsVersion
- GetSearchHitResultsRevision

- GetSearchHitResultsVersion
- DestroySearch
- SearchWithResultsRevision
- SearchWithResultsVersion

3.4.1 Search

The method to execute a search for documents has the following signature:

```
int Search (Credentials credentials,
            string searchTemplateName,
            MetadataItem[] query,
            out Guid searchID);
```

credentials	The credentials to verify the calling user for this call.
searchTemplateName	The name of the Search Template used to run this search.
query	The criteria to be used when performing the search.
out searchID	The unique identifier of the executed search will be available in this parameter after searching is complete.
return int	The number of results found from the search.

This method only executes the search, no results will be returned. To get results from this search, call *GetSearchResultsRevision* or *GetSearchResultsVersion* or *GetSearchHitResultsRevision* or *GetSearchHitResultsVersion* using the unique identifier returned in the searchID parameter in this call. Once you have finished with a search, *DestroySearch* should be called to clear up resources.

3.4.2 GetSearchResults{Revision|Version}

The method to get revision identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
string[] GetSearchResultsRevision (Credentials credentials,
                                   Guid searchID,
                                   int index,
                                   int count,
                                   out MetadataItem[][] results);
```

The method to get versioned document identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
Guid[] GetSearchResultsVersion (Credentials credentials,
                                Guid searchID,
                                int index,
                                int count,
                                out MetadataItem[][] results,
                                out string[] documentVersions);
```

credentials	The credentials to verify the calling user for this call.
searchID	The unique identifier of the search used to retrieve results.
index	The index of the first result to retrieve (0-based).
count	The number of results to retrieve from the starting index.
out results	The document metadata requested from the search will be available in this parameter after retrieval is complete. Note that only a single index set is returned for each document.
out documentVersions	The list of document version numbers will be placed into this array. Each index links to the corresponding index in the results array.
return string[]	An array of unique revision identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.
return Guid[]	An array of unique document identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.

This method returns the up-to-date document metadata (excluding search assistant data) found in the search results. Any metadata changes performed between search execution and results retrieval will show. If a document has been deleted, its unique reference will still be visible in the search results but its metadata will be *null*.

3.4.3 GetSearchHitResults{Revision|Version}

The method to get revision identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
string[] GetSearchHitResultsRevision (Credentials credentials,
                                      Guid searchID,
                                      int index,
                                      int count,
                                      out MetadataItem[][] results);
```

The method to get versioned document identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
Guid[] GetSearchHitResultsVersion (Credentials credentials,
                                   Guid searchID,
                                   int index,
                                   int count,
                                   out MetadataItem[][] results,
                                   out string[] documentVersions);
```

credentials	The credentials to verify the calling user for this call.
searchID	The unique identifier of the search used to retrieve results.
index	The index of the first result to retrieve (0-based).

count	The number of results to retrieve from the starting index.
out results	The search results metadata (including search assistant data) requested from the search will be available in this parameter after retrieval is complete. Note that only a single index set is returned for each document.
out documentVersions	The list of document version numbers will be placed into this array. Each index links to the corresponding index in the results array.
return string[]	An array of unique revision identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.
return Guid[]	An array of unique document identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.

This method returns the search results metadata (including search assistant data) found when the search was performed. Any metadata changes performed between search execution and results retrieval will not be reflected in these results.

3.4.4 DestroySearch

The method to destroy a previously executed search has the following signature:

```
void DestroySearch (Credentials credentials,
                    Guid searchID);
```

credentials	The credentials to verify the calling user for this call.
searchID	The unique identifier of the search used to retrieve results.
return void	Nothing.

Once a user has finished with a search, this method should be called to ensure good resource management, but it will be called after a customer specified period by the DataStoreDSX housekeeping process if the third party system fails to do so.

3.4.5 SearchWithResults{Revision|Version}

The method to execute a search for documents and return all the results has the following signature:

```
int SearchWithResultsRevision (Credentials credentials,
                               string searchTemplateName,
                               MetadataItem[] query,
                               out Guid searchID,
                               out string[] revisionIDs,
                               out MetadataItem[][] results);
```

The method to execute a search for documents and return all the results with document identifiers and versions has the following signature:

```
int SearchWithResultsVersion (Credentials credentials,
                              string searchTemplateName,
                              MetadataItem[] query,
                              out Guid searchID,
                              out Guid[] documentIDs,
                              out string[] documentVersions,
                              out MetadataItem[][] results);
```

credentials	The credentials to verify the calling user for this call.
searchTemplateName	The name of the search template in which to run this search against.
query	The criteria to be used when performing the search.
out searchID	The unique identifier of the executed search will be available in this parameter after searching is complete.
out revisionIDs	An array of unique revision identifiers for the documents relating to each search result will be available in this parameter after the search is complete. Each index links to the corresponding index in the results array.
out documentIDs	An array of unique document identifiers for the documents relating to each search result will be available in this parameter after the search is complete. Each index links to the corresponding index in the results array.
out documentVersions	The versions of each of the documents. Each index links to the corresponding value in the documentIDs array.
out results	The search results metadata (including search assistant data) requested from the search will be available in this parameter after retrieval is complete. Note that only a single index set is returned for each document.
return int	The number of results found from the search.

This method performs *Search*, *GetSearchHitResults{Revision|Version}* (returning all data found) and *DestroySearch*. Once this call has been executed, the search is no longer retrievable as it has been destroyed.

3.5 Updating Document Data

The following methods are available that are associated with the updating of document data:

- UpdateDocumentMetadataRevision
- UpdateDocumentMetadataSetsRevision
- UpdateDocumentMetadataVersion
- UpdateDocumentMetadataSetsVersion

3.5.1 UpdateDocumentMetadata[Sets]{Revision|Version}

The methods to update document details in DataStoreDSX have the following signatures:

```
void UpdateDocumentMetadataRevision (Credentials credentials,
                                     string revisionID,
                                     MetadataItem[] metadata);
```

```
void UpdateDocumentMetadataSetsRevision (Credentials credentials,
                                         string revisionID,
                                         MetadataItem[][] metadata);
```

The methods to update document details for the specific version of a document in DataStoreDSX have the following signatures:

```
void UpdateDocumentMetadataVersion (Credentials credentials,
                                    Guid documentID,
                                    string documentVersion,
                                    MetadataItem[] metadata);
```

```
void UpdateDocumentMetadataSetsVersion (Credentials credentials,
                                        Guid documentID,
                                        string documentVersion,
                                        MetadataItem[][] metadata);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to which the new metadata is related.
documentID	The unique document identifier of the document to which the new metadata is related.
documentVersion	The version number of the document.
metadata	The new metadata for the document. Method names without the word "Sets" in them accept a single dimension array containing a single replacement index set, while method names with the word "Sets" in them accept a two dimensional array containing one or more replacement index sets.
return void	Nothing.

This update treats metadata as a complete replacement. All original fields are removed and the new data inserted. Any field values not present in the new data are not retained.

3.6 Moving Documents

The following methods are available that allow documents to be moved to different data definitions:

- MoveDocumentRevision

- MoveDocumentSetsRevision
- MoveDocumentVersion
- MoveDocumentSetsVersion

3.6.1 MoveDocument[Sets]{Revision|Version}

The methods to move documents to different data definitions in DataStoreDSX have the following signatures:

```
void MoveDocumentRevision (Credentials credentials,
                           string revisionID,
                           string newDataDefinitionName,
                           MetadataItem[] newMetadata);
```

```
void MoveDocumentSetsRevision (Credentials credentials,
                               string revisionID,
                               string newDataDefinitionName,
                               MetadataItem[][] newIndexSets);
```

The methods to move documents to different data definitions for a specific version of a document in DataStoreDSX have the following signatures:

```
void MoveDocumentVersion (Credentials credentials,
                          Guid documentID,
                          string documentVersion,
                          string newDataDefinitionName,
                          MetadataItem[] newMetadata);
```

```
void MoveDocumentSetsVersion (Credentials credentials,
                              Guid documentID,
                              string documentVersion,
                              string newDataDefinitionName,
                              MetadataItem[][] newIndexSets);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to move to a new data definition.
documentID	The unique document identifier of the document to move to a new data definition.
documentVersion	The version number of the document.
newDataDefinitionName	The name of the data definition into which to move the document.
newMetadata	The new metadata for the document, as a single dimension array representing a single index set.
newIndexSets	The new metadata for the document, as a two dimensional array representing one or more replacement index sets.

return void	Nothing.

This movement treats metadata as a complete replacement. All original fields are removed and the new data inserted. Any field values not present in the new data are not retained.

3.7 Deleting Documents

The following available methods are associated with the deletion of document data:

- DeleteDocumentRevision
- DeleteDocumentVersion

3.7.1 DeleteDocument{Revision|Version}

The method to delete a document from DataStoreDSX has the following signature:

```
void DeleteDocumentRevision (Credentials credentials,
                             string revisionID);
```

The method to delete all versions of a document from DataStoreDSX has the following signature:

```
void DeleteDocumentVersion (Credentials credentials,
                            Guid documentID);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to delete.
documentID	The unique document identifier of the document to delete.
return void	Nothing.

Once a document has been deleted, its metadata and contents will be removed from DataStoreDSX and cannot be undone.

3.8 Deprecated Methods

The following methods have been deprecated. Where possible, please upgrade calling code to invoke the alternatives.

Old Method Name	**New Method Name**
StoreNewDocument	StoreNewRevisionedDocument Assign udi to revisionID parameter.
StoreNewDocumentSets	StoreNewRevisionedDocumentSets Assign udi to revisionID parameter.
GetDocument	GetDocumentRevision Assign udi to revisionID parameter.

GetDocumentMetadata	GetDocumentMetadataRevision Assign udi to revisionID parameter.
GetDocumentMetadataSets	GetDocumentMetadataSetsRevision Assign udi to revisionID parameter.
GetDocumentProperties	GetDocumentPropertiesRevision Assign udi to revisionID parameter.
SearchWithResults	SearchWithResultsRevision Assign udi to revisionID parameter.
GetSearchResults	GetSearchResultsRevision Assign udi to revisionID parameter.
GetSearchHitResults	GetSearchHitResultsRevision Assign udi to revisionID parameter.
UpdateDocumentMetadata	UpdateDocumentMetadataRevision Assign udi to revisionID parameter.
UpdateDocumentMetadataSets	UpdateDocumentMetadataSetsRevision Assign udi to revisionID parameter.
DeleteDocument	DeleteDocumentRevision Assign udi to revisionID parameter.

3.9 Recent Changes

In all previous versions of the API, a generic exception would be thrown whenever it was determined that a document did not exist. As of the latest version of the API, an exception of type FaultException<DocumentDoesNotExist> is thrown instead.

This makes it possible to know that a document does not exist based on the exception type rather than the exception message.

The only caveat is that the old behaviour takes place whenever an attempt is made to delete a document that does not exist when specifying a revision identifier rather than a version identifier.

4 Document Handling COM-compatible API

The Document Handling COM-compatible API provides third parties with direct access to a DataStoreDSX system through a COM-compatible interface.

All methods inside the API are synchronous, and will block until completion.

Should an error occur, detailed error information is returned by the API.

4.1 COM API - Complex Data Types

The COM-based DSX API makes use of the following complex data types.

4.1.1 Credentials

Each method takes in a *Credentials* object which has two known types: *CASCredentials* and *UserNameCredentials*. If null is passed in, the service will assume that Windows Authentication is being used.

4.1.1.1 CasCredentials

Represents Central Authentication Service (CAS) credentials, which can be passed into methods called for authentication purposes.

Member	Data Type	Comments
ServiceTicket	string	The CAS service ticket.
TicketGrantingTicket	string	The CAS ticket granting ticket.

4.1.1.2 UserNameCredentials

Represents username-based credentials, which can be passed into methods called for authentication purposes.

Member	Data Type	Comments
UserName	string	The user name of the user.
Password	string	The user's password.

4.1.2 MetadataItem

A specific piece of metadata associated with a document which can be used to pass metadata into DataStoreDSX for searching, storing and updating purposes. It is also used to retrieve metadata from DataStoreDSX. A *MetadataItem* consists of a field name and value. The field name is the name given to the custom field type, which must be provided for each *MetadataItem*. The value can be of type *string*, *DateTime*, *integer*, or *decimal* (as appropriate to the Custom Field Type).

Member	Data Type	Comments

FieldName	string	The name of the metadata field.
Value	List<object>	The list of values of the metadata field.

Note that the MetadataItem type contains the following methods for composing or analysing a set of values associated with a metadata item.

Methods that allow values to be added to the list:

```
void AddInt32([in] long value);
void AddDecimal([in] CURRENCY value);
void AddString([in] BSTR value);
void AddDate([in] DATE value);
```

The method that allows the number of values in the list to be retrieved:

```
long GetValueCount();
```

The method that allows a specific value to be retrieved, based on its index in the list (note that the value is converted to a string first):

```
BSTR GetValue([in] long index);
```

4.1.3 IndexSet

Represents a collection of metadata associated with a document, which comprises a single index set. This allows various methods to accept or return multiple document index sets.

Member	Data Type	Comments
Metadata	List<MetadataItem>	The collection of document metadata items representing a single index set.

The following methods allow maintenance of metadata items present in specific index sets:

```
SAFEARRAY(IMetadataItem*) GetMetadata();
void AddMetadataItem([in] IMetadataItem* item);
```

4.1.4 SearchResult

Represents a single result returned by a search operation. It contains a single method for retrieving the list of metadata items that make up the search result:

```
SAFEARRAY(IMetadataItem*) GetMetadata();
```

4.2 Storing a New Document

The following available methods are associated with storing new document data:

- StoreNewRevisionedDocument
- StoreNewRevisionedDocumentSets
- StoreNewVersionedDocument
- StoreNewVersionedDocumentSets
- StoreNewVersionedDocumentIdentifier
- StoreNewVersionedDocumentSetsIdentifier
- StoreNewVersionedDocumentVersion
- StoreNewVersionedDocumentSetsVersion

4.2.1 StoreNewRevisionedDocument[Sets]

The methods to store a new document and its associated metadata, returning a document revision identifier, have the following signatures:

```
void StoreNewRevisionedDocument(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata,
                [in, out] BSTR* revisionID);
```

```
void StoreNewRevisionedDocumentSets(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
                [in, out] SAFEARRAY(IIndexSet*)* indexSets,
                [in, out] BSTR* revisionID);
```

in credentials	The credentials for verifying the calling user.
in dataDefinitionName	The name of the data definition which is used to store the new document.
in, out document	The document contents.
in, out metadata	The metadata to associate with the document, which represents a single index set.
in, out indexSets	The metadata to associate with the document, which represents one or more index sets.
in, out revisionID	The unique revision identifier assigned by DataStoreDSX when the document is stored.

4.2.2 StoreNewVersionedDocument[Sets][Version|Identifier]

The methods to store a new specifically versioned document and its associated metadata have the following signatures:

```
void StoreNewVersionedDocument(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata,
                [in, out] BSTR* documentID,
                [in, out] BSTR* documentVersion);
```

```
void StoreNewVersionedDocumentSets(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
```

```
                [in, out] SAFEARRAY(IIndexSet*)* indexSets,
                [in, out] BSTR* documentID,
                [in, out] BSTR* documentVersion);
```

```
void StoreNewVersionedDocumentIdentifier(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata,
                [in] BSTR documentID,
                [in, out] BSTR* documentVersion);
```

```
void StoreNewVersionedDocumentSetsIdentifier(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
                [in, out] SAFEARRAY(IIndexSet*)* indexSets,
                [in] BSTR documentID,
                [in, out] BSTR* documentVersion);
```

```
void StoreNewVersionedDocumentVersion(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata,
                [in] BSTR documentID,
                [in] BSTR documentVersion);
```

```
void StoreNewVersionedDocumentSetsVersion(
                [in] ICredentials* credentials,
                [in] BSTR dataDefinitionName,
                [in, out] SAFEARRAY(unsigned char)* document,
                [in, out] SAFEARRAY(IIndexSet*)* indexSets,
                [in] BSTR documentID,
                [in] BSTR documentVersion);
```

in credentials	The credentials for verifying the calling user.
in dataDefinitionName	The name of the data definition which is used to store the new document.
in, out document	The document contents.
in, out metadata	The metadata to associate with the document, which represents a single index set.
in, out indexSets	The metadata to associate with the document, which represents one or more index sets.

in revisionID or **in, out** revisionID	The unique revision identifier. Depending on the method being called, it is either assigned by the caller or assigned by DataStoreDSX.
in documentID or **in, out** documentID	The unique document identifier. Depending on the method being called, it is either assigned by the caller or assigned by DataStoreDSX.
in documentVersion or **in, out** documentVersion	The version number of the document. Depending on the method being called, it is either assigned by the caller or assigned by DataStoreDSX.

4.3 Retrieving Document Data

The following available methods are associated with the retrieval of document data:

- GetDocumentRevision
- GetDocumentVersion
- GetDocumentMetadataRevision
- GetDocumentMetadataSetsRevision
- GetDocumentMetadataVersion
- GetDocumentMetadataSetsVersion
- GetDocumentPropertiesRevision
- GetDocumentPropertiesVersion

4.3.1 GetDocument{Revision|Version}

The method to retrieve the contents of the document has the following signature:

```
void GetDocumentRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in] BSTR mimeType,
                [in, out] SAFEARRAY(long)* pages,
                [in, out] SAFEARRAY(unsigned char)* data);
```

The method to retrieve the contents of the document based on document identifier and version has the following signature:

```
void GetDocumentVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in] BSTR mimeType,
                [in, out] SAFEARRAY(long)* pages,
                [in, out] SAFEARRAY(unsigned char)* data);
```

in credentials	The credentials for verifying the calling user.

in revisionID	The unique revision identifier of the document to be retrieved from DataStoreDSX.
in documentID	The unique document identifier of the document to be retrieved from DataStoreDSX.
in documentVersion	The version number of the document.
in mimeType	The mime type to return the document as. Pass as null to return the document in its stored format.
in, out pages	The individual pages to retrieve from the document (if applicable e.g. 1, 3, 4 returns pages 1, 3 and 4). Pass as null to return all pages.
in, out data	The contents of the document will be available in this parameter after retrieval is complete.

4.3.2 GetDocumentMetadata[Sets]{Revision|Version}

The methods to retrieve document metadata have the following signatures:

```
void GetDocumentMetadataRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata);
```

```
void GetDocumentMetadataSetsRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in, out] SAFEARRAY(IIndexSet*)* indexSets);
```

The methods to retrieve document metadata from a specific version of a document have the following signatures:

```
void GetDocumentMetadataVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata);
```

```
void GetDocumentMetadataSetsVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in, out] SAFEARRAY(IIndexSet*)* indexSets);
```

in credentials	The credentials for verifying the calling user.

in revisionID	The unique revision identifier of the document for which to retrieve metadata.
in documentID	The unique document identifier of the document for which to retrieve metadata.
in documentVersion	The version number of the document.
in, out metadata	The metadata of the document will be available in this parameter after retrieval is complete. The metadata is the document's first index set.
in, out indexSets	The metadata of the document will be available in this parameter after retrieval is complete. The metadata is comprises all of the document's available index sets.

4.3.3 GetDocumentProperties{Revision|Version}

The methods to retrieve document properties have the following signatures:

```
void GetDocumentPropertiesRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata);
```

```
void GetDocumentPropertiesRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata,
                [in, out] BSTR* mimeType);
```

The methods to retrieve document properties based on the version of a document have the following signatures:

```
void GetDocumentPropertiesVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata);
```

```
void GetDocumentPropertiesVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata,
                [in, out] BSTR* mimeType);
```

in credentials	The credentials for verifying the calling user.
in revisionID	The unique revision identifier of the document for which to retrieve properties.
in documentID	The unique document identifier of the document for which to retrieve properties.
in documentVersion	The version number of the document.
in, out metadata	The properties of the document will be available in this parameter after retrieval is complete.
in, out mimeType	The format of the specified document.

4.3.4 GetDocumentVersionHistory

The method to retrieve version numbers associated with a document has the following signature:

```
void GetDocumentVersionHistory(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in, out] SAFEARRAY(BSTR)* documentVersions);
```

credentials	The credentials to verify the calling user for this call.
documentID	The unique document identifier of the document whose version numbers to retrieve.
in, out documentVersions	Returns the list of version numbers associated with the specified document.

4.4 Searching Document Data

The following available methods are associated with the searching of document data:

- Search
- GetSearchResultsRevision
- GetSearchResultsVersion
- GetSearchHitResultsRevision
- GetSearchHitResultsVersion
- DestroySearch
- SearchWithResultsRevision
- SearchWithResultsVersion

4.4.1 Search

The method to execute a search for documents has the following signature:

```
long Search(
                [in] ICredentials* credentials,
                [in] BSTR searchTemplateName,
                [in, out] SAFEARRAY(IMetadataItem*)* query,
                [in, out] BSTR* searchID);
```

in credentials	The credentials for verifying the calling user.
in searchTemplateName	The name of the Search Template used to run this search.
in, out query	The criteria to be used when performing the search.
in, out searchID	The unique identifier of the executed search will be available in this parameter after searching is complete.
result	The number of results found from the search.

4.4.2 GetSearchResults{Revision|Version}

The method to get revision identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
void GetSearchResultsRevision(
                [in] ICredentials* credentials,
                [in] BSTR searchID,
                [in] long index,
                [in] long count,
                [in, out] SAFEARRAY(BSTR)* revisionIDs,
                [in, out] SAFEARRAY(ISearchResult*)* results);
```

The method to get document identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
void GetSearchResultsVersion(
                [in] ICredentials* credentials,
                [in] BSTR searchID,
                [in] long index,
                [in] long count,
                [in, out] SAFEARRAY(BSTR)* documentIDs,
                [in, out] SAFEARRAY(BSTR)* documentVersions,
                [in, out] SAFEARRAY(ISearchResult*)* results);
```

in credentials	The credentials for verifying the calling user.
in searchID	The unique identifier of the search used to retrieve results.
in index	The index of the first result to retrieve (0-based).
in count	The number of results to retrieve from the starting index.

in, out revisionIDs	An array of unique revision identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.
in, out documentIDs	An array of unique document identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.
in, out documentVersions	The version number of each document. Each index links to the corresponding index in the results array.
in, out results	The document metadata requested from the search will be available in this parameter after retrieval is complete.

This method returns the up-to-date document metadata (not search assistant data) found in the search results. Any metadata changes performed between search execution and results retrieval will show. If a document has been deleted, its unique reference will still be visible in the search results but its metadata will be *null*.

4.4.3 GetSearchHitResults{Revision|Version}

The method to get revision identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
void GetSearchHitResultsRevision(
                [in] ICredentials* credentials,
                [in] BSTR searchID,
                [in] long index,
                [in] long count,
                [in, out] SAFEARRAY(BSTR)* revisionIDs,
                [in, out] SAFEARRAY(ISearchResult*)* results);
```

The method to get document identifier based search results from a previously executed undestroyed search using document metadata has the following signature:

```
void GetSearchHitResultsVersion(
                [in] ICredentials* credentials,
                [in] BSTR searchID,
                [in] long index,
                [in] long count,
                [in, out] SAFEARRAY(BSTR)* documentIDs,
                [in, out] SAFEARRAY(BSTR)* documentVersions,
                [in, out] SAFEARRAY(ISearchResult*)* results);
```

in credentials	The credentials for verifying the calling user.
in searchID	The unique identifier of the search used to retrieve results.
in index	The index of the first result to retrieve (0-based).

in count	The number of results to retrieve from the starting index.
in, out revisionIDs	An array of unique revision identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.
in, out documentIDs	An array of unique document identifiers for the documents relating to each search result. Each index links to the corresponding index in the results array.
in, out documentVersions	The version of each document. Each index links to the corresponding index in the results array.
in, out results	The search results metadata (including search assistant data) requested from the search will be available in this parameter after retrieval is complete.

This method returns the search results metadata (including search assistant data) found when the search was performed. Any metadata changes performed between search execution and results retrieval will not be reflected in these results.

4.4.4 DestroySearch

The method to destroy a previously executed search has the following signature:

```
void DestroySearch(
                [in] ICredentials* credentials,
                [in] BSTR searchID);
```

in credentials	The credentials for verifying the calling user.
in searchID	The unique identifier of the search used to retrieve results.

4.4.5 SearchWithResults{Revision|Version}

The method to execute a search for documents and return all the results has the following signature:

```
long SearchWithResultsRevision(
                [in] ICredentials* credentials,
                [in] BSTR searchTemplateName,
                [in, out] SAFEARRAY(IMetadataItem*)* query,
                [in, out] BSTR* searchID,
                [in, out] SAFEARRAY(BSTR)* revisionIDs,
                [in, out] SAFEARRAY(ISearchResult*)* results);
```

The method to execute a search for documents and return all the results with document identifiers and versions has the following signature:

```
long SearchWithResultsVersion(
                [in] ICredentials* credentials,
                [in] BSTR searchTemplateName,
                [in, out] SAFEARRAY(IMetadataItem*)* query,
                [in, out] BSTR* searchID,
                [in, out] SAFEARRAY(BSTR)* documentIDs,
                [in, out] SAFEARRAY(BSTR)* documentVersions,
                [in, out] SAFEARRAY(ISearchResult*)* results);
```

in credentials	The credentials for verifying the calling user.
in searchTemplateName	The name of the search template in which to run this search against.
in, out query	The criteria to be used when performing the search.
in, out searchID	The unique identifier of the executed search will be available in this parameter after searching is complete.
in, out revisionIDs	An array of unique revision identifiers for the documents relating to each search result will be available in this parameter after the search is complete. Each index links to the corresponding index in the results array.
in, out documentIDs	An array of unique document identifiers for the documents relating to each search result will be available in this parameter after the search is complete. Each index links to the corresponding index in the results array.
in, out documentVersions	The versions of each of the documents. Each index links to the corresponding index in the results array.
in, out results	The search results metadata (including search assistant data) requested from the search will be available in this parameter after retrieval is complete.
result	The number of results found from the search.

This method performs *Search*, *GetSearchHitResults{Revision|Version}* (returning all data found) and *DestroySearch*. Once this call has been executed, the search is no longer retrievable as it has been destroyed.

4.5 Updating Document Data

The following available methods are associated with the updating of document data:

- UpdateDocumentMetadataRevision
- UpdateDocumentMetadataSetsRevision
- UpdateDocumentMetadataVersion
- UpdateDocumentMetadataSetsVersion

4.5.1 UpdateDocumentMetadata[Sets]{Revision|Version}

The methods to update document details in DataStoreDSX have the following signatures:

```
void UpdateDocumentMetadataRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata);
```

```
void UpdateDocumentMetadataSetsRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in, out] SAFEARRAY(IIndexSet*)* indexSets);
```

The methods to update document details for the specific version of a document in DataStoreDSX have the following signatures:

```
void UpdateDocumentMetadataVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in, out] SAFEARRAY(IMetadataItem*)* metadata);
```

```
void UpdateDocumentMetadataSetsVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in, out] SAFEARRAY(IIndexSet*)* indexSets);
```

in credentials	The credentials for verifying the calling user.
in revisionID	The unique revision identifier of the document to which the new metadata is related.
in documentID	The unique document identifier of the document to which the new metadata is related.
in documentVersion	The version number of the document.
in, out metadata	The new metadata for the document, as a single index set.
in, out indexSets	The new metadata for the document, comprising one or more index sets.

This update treats metadata as a complete replacement. All original fields are removed and the new data inserted. Any field values not present in the new data are not retained.

4.6 Moving Documents

The following methods are available that allow documents to be moved to different data definitions:

- MoveDocumentRevision
- MoveDocumentSetsRevision
- MoveDocumentVersion
- MoveDocumentSetsVersion

4.6.1 MoveDocument[Sets]{Revision|Version}

The methods to move documents to different data definitions in DataStoreDSX have the following signatures:

```
void MoveDocumentRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in] BSTR newDataDefinitionName,
                [in, out] SAFEARRAY(IMetadataItem*)* newMetadata);
```

```
void MoveDocumentSetsRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID,
                [in] BSTR newDataDefinitionName,
                [in, out] SAFEARRAY(IIndexSet*)* newIndexSets);
```

```
void MoveDocumentVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in] BSTR newDataDefinitionName,
                [in, out] SAFEARRAY(IMetadataItem*)* newMetadata);
```

```
void MoveDocumentSetsVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID,
                [in] BSTR documentVersion,
                [in] BSTR newDataDefinitionName,
                [in, out] SAFEARRAY(IIndexSet*)* newIndexSets);
```

in credentials	The credentials to verify the calling user for this call.
in revisionID	The unique revision identifier of the document to move to a new data definition.
in documentID	The unique document identifier of the document to move to a new data definition.
in documentVersion	The version number of the document.

in newDataDefinitionName	The name of the data definition into which to move the document.
in, out newMetadata	The new metadata for the document, as a single dimension array representing a single index set.
in, out newIndexSets	The new metadata for the document, as a two dimensional array representing one or more replacement index sets.

This movement treats metadata as a complete replacement. All original fields are removed and the new data inserted. Any field values not present in the new data are not retained.

4.7 Deleting Documents

The following available methods are associated with the deletion of document data:

- DeleteDocumentRevision
- DeleteDocumentVersion

4.7.1 DeleteDocument{Revision|Version}

The method to delete a document from DataStoreDSX has the following signature:

```
void DeleteDocumentRevision(
                [in] ICredentials* credentials,
                [in] BSTR revisionID);
```

The method to delete all versions of a document from DataStoreDSX has the following signature:

```
void DeleteDocumentVersion(
                [in] ICredentials* credentials,
                [in] BSTR documentID);
```

in credentials	The credentials for verifying the calling user.
in revisionID	The unique revision identifier of the document to delete.
in documentID	The unique document identifier of the document to delete.

Once a document has been deleted, its metadata and contents will be removed from DataStoreDSX and cannot be undone.

4.8 Deprecated Methods

The following methods have been deprecated. Where possible, please upgrade calling code to invoke the alternatives.

Old Method Name	New Method Name
StoreNewDocument	StoreNewRevisionedDocument

	Assign udi to revisionID parameter.
GetDocument	GetDocumentRevision Assign udi to revisionID parameter.
GetDocumentMetadata	GetDocumentMetadataRevision Assign udi to revisionID parameter.
GetDocumentProperties	GetDocumentPropertiesRevision Assign udi to revisionID parameter.
SearchWithResults	SearchWithResultsRevision Assign udi to revisionID parameter.
GetSearchResults	GetSearchResultsRevision Assign udi to revisionID parameter.
GetSearchHitResults	GetSearchHitResultsRevision Assign udi to revisionID parameter.
UpdateDocumentMetadata	UpdateDocumentMetadataRevision Assign udi to revisionID parameter.
DeleteDocument	DeleteDocumentRevision Assign udi to revisionID parameter.

4.9 Recent Changes

In all previous versions of the API, a generic exception would be thrown whenever it was determined that a document did not exist. As of the latest version of the API, an exception of type DocumentDoesNotExistException is thrown instead.

This makes it possible to know that a document does not exist based on the exception type rather than the exception message.

The only caveat is that the old behaviour takes place whenever an attempt is made to delete a document that does not exist when specifying a revision identifier rather than a version identifier.

5 Document Handling .NET API

The Document Handling .NET API is a C# class library that provides third parties with direct access to a DataStoreDSX system.

This API provides the ability to use DataStoreDSX to store, retrieve, search, update and delete data items.

All methods inside the API are synchronous, and will block until completion.

Exceptions are thrown by the API whenever errors occur.

5.1 .NET API - Complex Data Types

The document handling .NET API makes use of the following complex data types.

5.1.1 Credentials

Each method takes in a *Credentials* object which has two known types: *CasCredentials* and *UserNameCredentials*. If null is passed in, the service will assume that Windows Authentication is being used.

5.1.1.1 CasCredentials

Member	Data Type	Mandatory	Comments
ServiceTicket	string	Yes	The CAS service ticket
TicketGrantingTicket	string	Yes	The CAS ticket granting ticket

5.1.1.2 UserNameCredentials

Member	Data Type	Mandatory	Comments
UserName	string	Yes	The user name of the user
Password	string	Yes	The password of the user

5.1.2 MetadataItem

The *MetadataItem* can be used to pass metadata into DataStoreDSX for searching, storing and updating purposes. It is also used to retrieve metadata from DataStoreDSX. A *MetadataItem* consists of a field name and value. The field name is the name given to the custom field type, which must be provided for each *MetadataItem*. The value can be of type *string*, *DateTime*, *integer*, or *decimal* (as appropriate to the Custom Field Type).

Member	Data Type	Mandatory	Comments
FieldName	string	Yes	The name of the field in DataStoreDSX
Value	object[]	No	The list of values of the metadata field

5.1.3 IndexSet

An index sets contains a single set of metadata. This allows multiple index sets to be specified or returned.

Member	Data Type	Mandatory	Comments
Metadata	MetadataItem[]	Yes	The collection of document metadata items representing a single index set.

5.1.4 DocumentProperties

The document properties objects consist of the following information:

Member	Data Type	Comments
MimeType	string	The document's mime type
Properties	DocumentPropertyItem[]	The properties associated with the document.

5.1.5 DocumentPropertyItem

The document property item object has a name and value for a specific document property:

Member	Data Type	Comments
Name	string	The property name.
Value	object	The value of the property.

5.2 DocumentReference

A document reference permits the use of versioned document identifiers rather than document revision identifiers when specifying a reference to a document.

Member	Data Type	Mandatory	Comments
DocumentID	Guid	Yes	The document ID
Version	string	No	The version number of the document

5.3 SearchDetails

Contains information about a search that has been performed.

Member	Data Type	Comments
SearchID	Guid	The search ID of the search that was performed
NumberOfResults	int	The number of results that have been returned by the search

5.4 SearchResults

Contains a collection of search results.

Member	Data Type	Comments
SearchID	Guid	The search ID of the search that was performed
Results	SearchResult[]	The collection of search results

5.5 SearchResult

Contains information pertinent to a specific single search result.

Member	Data Type	Comments
RevisionID	string	The document revision identifier, which will have a value if DocumentReference is null
DocumentReference	DocumentReference	The versioned document reference, which will have a value if the revision identifier is not present
Metadata	MetadataItem[]	The first index set of the found document as a collection of metadata items

5.6 Storing a New Document

The following available methods are associated with storing new document data:

- StoreNewDocument
- StoreNewDocumentVersion

5.6.1 StoreNewDocument[Version]

The method to store a new document and its associated metadata have the following signatures:

```
string StoreNewDocument (Credentials credentials,
                         string dataDefinitionName,
                         byte[] document,
                         MetadataItem[] metadata);
```

```
string StoreNewDocument (Credentials credentials,
                         string dataDefinitionName,
                         byte[] document,
                         IndexSet[] indexSets);
```

The methods to store a new document and its associated metadata returning a document identifier and version have the following signatures:

```
DocumentReference StoreNewDocumentVersion (Credentials credentials,
                                           string dataDefinitionName,
                                           byte[] document,
                                           MetadataItem[] metadata);
```

```
DocumentReference StoreNewDocumentVersion (Credentials credentials,
                                           string dataDefinitionName,
                                           byte[] document,
                                           IndexSet[] indexSets);
```

```
DocumentReference StoreNewDocumentVersion (Credentials credentials,
                                           string dataDefinitionName,
                                           byte[] document,
                                           MetadataItem[] metadata,
                                           Guid documentID,
                                           string documentVersion = null);
```

```
DocumentReference StoreNewDocumentVersion (Credentials credentials,
                                           string dataDefinitionName,
                                           byte[] document,
                                           IndexSet[] indexSets,
                                           Guid documentID,
                                           string documentVersion = null);
```

credentials	The credentials to verify the calling user for this call.
dataDefinitionName	The name of the Data Definition which is used to store the new document.
document	The documents contents to store in DataStoreDSX.
metadata	The metadata to associate with this document, as a single index set.
indexSets	The metadata to associate with this document, as one or more index sets.
documentID	The document identifier to assign to the newly stored document. If not specified, then the document identifier is determined by DataStoreDSX.
documentVersion	The version number to assign to the newly stored document. If not specified, then the version number is determined by DataStoreDSX.
return string	The unique revision identifier assigned by DataStoreDSX when the document is stored.
return DocumentReference	The unique document identifier assigned by DataStoreDSX when the document is stored, which is a versioned document reference.

5.7 Retrieving Document Data

The following available methods are associated with the retrieval of document data:

- GetDocument

- GetDocumentMetadata
- GetDocumentMetadataSets
- GetDocumentProperties

5.7.1 GetDocument

The method to retrieve the contents of the document has the following signature:

```
byte[] GetDocument (Credentials credentials,
                    string revisionID,
                    string mimeType,
                    int[] pages);
```

The method to retrieve the contents of the document based on document identifier and version has the following signature:

```
byte[] GetDocument (Credentials credentials,
                    DocumentReference reference,
                    string mimeType,
                    int[] pages);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to be retrieved from DataStoreDSX.
reference	The unique versioned identifier of the document to be retrieved from DataStoreDSX.
mimeType	The mime type to return the document as. Pass as null to return the document in its stored format.
pages	The individual pages to retrieve from the document (if applicable e.g. 1, 3, 4 returns pages 1, 3 and 4). Pass as null to return all pages.
return byte[]	The contents of the document.

5.7.2 GetDocumentMetadata[Sets]

The methods to retrieve document metadata have the following signatures:

```
MetadataItem[] GetDocumentMetadata (Credentials credentials,
                                    string revisionID);
```

```
IndexSet[] GetDocumentMetadataSets (Credentials credentials,
                                    string revisionID);
```

The methods to retrieve document metadata from a specific version of a document have the following signatures:

```
MetadataItem[] GetDocumentMetadata (Credentials credentials,
                                    DocumentReference reference);
```

```
IndexSet[] GetDocumentMetadataSets (Credentials credentials,
                                    DocumentReference reference);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document for which to retrieve metadata.
reference	The unique versioned identifier of the document for which to retrieve metadata.
return MetadataItem[]	The document metadata, which is its first index set.
return IndexSet[]	The document metadata, comprising all of its index sets.

5.7.3 GetDocumentProperties

The method to retrieve document properties has the following signature:

```
DocumentProperties GetDocumentProperties (Credentials credentials,
                                          string revisionID);
```

The method to retrieve document properties based on the version of a document has the following signature:

```
DocumentProperties GetDocumentProperties (Credentials credentials,
                                          DocumentReference reference);
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document for which to retrieve properties.
reference	The unique document identifier of the document for which to retrieve properties.
return DocumentProperties	The properties of the document.

5.7.4 GetDocumentVersionHistory

The method to retrieve version numbers associated with a document has the following signature:

```
string[] GetDocumentVersionHistory (Credentials credentials,
                                    Guid documentID);
```

credentials	The credentials to verify the calling user for this call.
documentID	The unique document identifier of the document whose version numbers to retrieve.

return string[]	Returns the list of version numbers associated with the specified document.

5.8 Searching Document Data

The following available methods are associated with the searching of document data:

- Search
- GetSearchResults
- GetSearchHitResults
- DestroySearch
- SearchWithResults

5.8.1 Search

The method to execute a search for documents has the following signature:

```
SearchDetails Search (Credentials credentials,
                      string searchTemplateName,
                      MetadataItem[] query);
```

credentials	The credentials to verify the calling user for this call.
searchTemplateName	The name of the Search Template used to run this search.
query	The criteria to be used when performing the search.
return SearchDetails	The details about the search, which includes the search unique identifier and the number of results found from the search.

This method only executes the search, no results will be returned. To get results from this search, either call *GetSearchResults* or *GetSearchHitResults* using the unique identifier returned in the searchID parameter in this call. Once you have finished with a search, *DestroySearch* should be called to clear up resources.

5.8.2 GetSearchResults

The method to get the search results from a previously executed undestroyed search using document metadata has the following signature:

```
SearchResult[] GetSearchResults (Credentials credentials,
                                 Guid searchID,
                                 int index,
                                 int count);
```

credentials	The credentials to verify the calling user for this call.
searchID	The unique identifier of the search used to retrieve results.
index	The index of the first result to retrieve (0-based).

count	The number of results to retrieve from the starting index.
return SearchResult[]	The requested search results, each of which contains each relevant document's metadata. Note that only a single index set is returned for each document.

This method returns the up-to-date document metadata (excluding search assistant data) found in the search results. Any metadata changes performed between search execution and results retrieval will show. If a document has been deleted, its unique reference will still be visible in the search results but its metadata will be *null*.

5.8.3 GetSearchHitResults

The method to get the search results from a previously executed undestroyed search using the search result data as the following signature:

```
SearchResult[] GetSearchHitResults (Credentials credentials,
                                    Guid searchID,
                                    int index,
                                    int count);
```

credentials	The credentials to verify the calling user for this call.
searchID	The unique identifier of the search used to retrieve results.
index	The index of the first result to retrieve (0-based).
count	The number of results to retrieve from the starting index.
return SearchResult[]	The requested search results (including search assistant data), each of which contains each relevant document's metadata. Note that only a single index set is returned for each document.

This method returns the search results metadata (including search assistant data) found when the search was performed. Any metadata changes performed between search execution and results retrieval will not be reflected in these results.

5.8.4 DestroySearch

The method to destroy a previously executed search has the following signature:

```
void DestroySearch (Credentials credentials,
                    Guid searchID);
```

credentials	The credentials to verify the calling user for this call.
searchID	The unique identifier of the search used to retrieve results.
return void	Nothing.

Once a user has finished with a search, this method should be called to ensure good resource management, but it will be called after a customer specified period by the DataStoreDSX housekeeping process if the third party system fails to do so.

5.8.5 SearchWithResults

The method to execute a search for documents and return all the results has the following signature:

```
SearchResults SearchWithResults (Credentials credentials,
                                 string searchTemplateName,
                                 MetadataItem[] query)
```

credentials	The credentials to verify the calling user for this call.
searchTemplateName	The name of the search template in which to run this search against.
query	The criteria to be used when performing the search.
return SearchResults	The search results (including the search ID, document references, document metadata and search assistant data). Note that only a single index set is returned for each document.

This method performs *Search*, *GetSearchHitResults* (returning all data found) and *DestroySearch*. Once this call has been executed, the search is no longer retrievable as it has been destroyed.

5.9 Updating Document Data

The following methods are available that are associated with the updating of document data:

- UpdateDocumentMetadata

5.9.1 UpdateDocumentMetadata

The methods to update document details in DataStoreDSX have the following signatures:

```
void UpdateDocumentMetadata (Credentials credentials,
                             string revisionID,
                             MetadataItem[] metadata)
```

```
void UpdateDocumentMetadata (Credentials credentials,
                             string revisionID,
                             IndexSet[] indexSets)
```

The methods to update document details for the specific version of a document in DataStoreDSX have the following signatures:

```
void UpdateDocumentMetadata (Credentials credentials,
                             DocumentReference reference,
                             MetadataItem[] metadata)
```

```
void UpdateDocumentMetadata (Credentials credentials,
                             DocumentReference reference,
```

```
                                IndexSet[] indexSets)
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to which the new metadata is related.
reference	The unique versioned identifier of the document to which the new metadata is related.
metadata	The new replacement metadata for the document, as a single index.
indexSets	The new replacement metadata for the document, as one or more index sets.
return void	Nothing.

This update treats metadata as a complete replacement. All original fields are removed and the new data inserted. Any field values not present in the new data are not retained.

5.10 Moving Documents

The following methods are available that allow documents to be moved to different data definitions:

- MoveDocument

5.10.1 MoveDocument

The methods to move documents to different data definitions in DataStoreDSX have the following signatures:

```
void MoveDocument (Credentials credentials,
                   string revisionID,
                   string newDataDefinitionName,
                   MetadataItem[] newMetadata);
```

```
void MoveDocument (Credentials credentials,
                   string revisionID,
                   string newDataDefinitionName,
                   IndexSet[] newIndexSets)
```

The methods to move documents to different data definitions for a specific version of a document in DataStoreDSX have the following signatures:

```
void MoveDocument (Credentials credentials,
                   DocumentReference reference,
                   string newDataDefinitionName,
                   MetadataItem[] newMetadata)
```

```
void MoveDocument (Credentials credentials,
                   DocumentReference reference,
```

```
                    string newDataDefinitionName,
                    IndexSet[] newIndexSets)
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to move to a new data definition.
reference	The unique versioned identifier of the document to move to a new data definition.
newDataDefinitionName	The name of the data definition into which to move the document.
newMetadata	The new metadata for the document, as a single replacement index set.
newIndexSets	The new metadata for the document, as one or more replacement index sets.
return void	Nothing.

This movement treats metadata as a complete replacement. All original fields are removed and the new data inserted. Any field values not present in the new data are not retained.

5.11 Deleting Documents

The following available methods are associated with the deletion of document data:

- DeleteDocument

5.11.1 DeleteDocument

The method to delete a document from DataStoreDSX has the following signature:

```
void DeleteDocument (Credentials credentials,
                     string revisionID)
```

The method to delete all versions of a document from DataStoreDSX has the following signature:

```
void DeleteDocument (Credentials credentials,
                     Guid documentID)
```

credentials	The credentials to verify the calling user for this call.
revisionID	The unique revision identifier of the document to delete.
documentID	The unique document identifier of the document to delete.
return void	Nothing.

Once a document has been deleted, its metadata and contents will be removed from DataStoreDSX and cannot be undone.

5.12 Deprecated Properties

The following properties have been deprecated. Where possible, please upgrade calling code to use the alternatives.

Old Property Name	New Property Name
SearchResult.Udi	SearchResults.RevisionID

5.13 Recent Changes

In all previous versions of the API, a generic exception would be thrown whenever it was determined that a document did not exist. As of the latest version of the API, an exception of type DocumentDoesNotExistException is thrown instead.

This makes it possible to know that a document does not exist based on the exception type rather than the exception message.

The only caveat is that the old behaviour takes place whenever an attempt is made to delete a document that does not exist when specifying a revision identifier rather than a version identifier.

6 Launching Indexing Studio via URL

Indexing Studio has the ability to be launched using a URL call to create an indexing session based on the data definition and the metadata passed in the URL.

The URL consists of the IP address of the hosting server and the port and website name, the ID of the data definition and the custom field type IDs of the data definition fields that should be populated.

The launching website will populate the fields using held values so values unchanged between sessions will be retained. If the previous values are not required then blank fields should be passed to clear the contents of the held values.

A sample URL is shown below:

```
http://[DataStoreDSX Server Name/IP Address]/indexingstudiolauncher/Default.aspx
?DataDefinition=[Data Definition ID (guid)]
&[Custom Field Type 1 ID (guid)]=999999999
&[Custom Field Type 2 ID (guid)]=123456789
&[Custom Field Type 3 ID (guid)]=11111111
&InitialOperation=None
```

The *InitialOperation* parameter defines the first operation that Indexing Studio will perform on launching. The options are *Scan*, *Import* or *None*. If *Scan* is supplied, Indexing Studio will scan a document from an attached scanner. If *Import* is supplied, Indexing Studio will display the *File Import* screen to allow the user to import a document. If *None* is supplied, Indexing Studio will not perform any specific operations upon launching. If the parameter is not supplied then the functionality will match that for *None*.

7 Launching Search Display Website via URL

The DataStoreDSX Search Display Website allows a search of the DSX system to be performed by passing the search criteria to the website. This will perform the search and return the results to a web site so the user can see a list of the documents found by the search and view documents by selecting them from the list. The search URL is shown below:

```
http://[DataStoreDSX Server Name/IP Address]/SearchDisplayWebsite/
?Template=Search%20Template
%20Name
&Search%20Field1=999999999
&Search%20Field2=123456789
```

The DataStoreDSX Search Display Website also provides the ability to display a document by calling the URL and passing a parameter of the DataStoreDSX unique document identifier for the required document. The format of this URL is shown below:

```
http://[DataStoreDSX Server Name/IP Address]/SearchDisplayWebsite/?UDI=[DSXUDI]
```

If the data item is not available, or is restricted from the user, or any other error occurs, an error message will be displayed to the user.

8 Search Parameter Syntax

The syntax used in Searching Client is described here.

8.1 Regular Search Field Syntax

The syntax used with Regular search fields, that is, search fields which are linked to Custom Field Types or to Index fields or to Audit fields, is explained below. (The syntax for Content Full Text search fields and Sub-search fields is described in Content Full Text Search Syntax and Sub-searches.

Note: Search parameter values are case sensitive is used to determine whether the search string is case sensitive.

8.1.1 Wildcard Characters

A wildcard character is used to represent one or more characters having 'any value'.

8.1.1.1 Text Fields

Wildcard operators in *Text* fields are described in Wildcard **Operators: Text Fields**.

Table 1. Wildcard Operators: Text Fields

Wildcard	Meaning
%	Zero or more characters. Example: • *Williams%* – any field values which start with Williams followed by zero or more characters such as *Williams, Williams!* and *Williamson*.
*	One or more characters. Example: • *Williams** – any field value that starts with *Williams* followed by one or more characters, such as *Williamson*, but *not* with *Williams*.
?	Exactly one character. Example: • *William?* – any field value that starts with *William* followed by one more characters, such as *Williams* and *William** but *not* with *Williamson*.

8.1.1.2 Date And Time Fields

Wildcard operators in *Date and Time* fields are described in Wildcard **Operators: Date and Time Fields**.

Table 2. Wildcard Operators: Date and Time Fields

Wildcard	Meaning
*	Wildcards the whole of the *Date and Time* field. Therefore, if the field has *Time* accuracy, the year, month, day and time are all wildcarded.
?	Parts of the *Date and Time* can be wildcarded as long as the *Year* has been specified. Examples (assuming a DD MM YYYY configuration for a *Time* accuracy field): • *?/?/2009 ?* – any time during 2009. • *1/10/2000 ?* – any time on the 1st October 2000 **Note:** It is *not possible* to wildcard the *year* of a *Time* Accuracy field. The text in the search field will turn red to indicate the syntax is invalid. For example 1/10/? ? (in a Time accuracy Date and Time field). Assuming DD MM YYYY configuration for a *Day* accuracy field: • *?/?/2009* – any day during 2009. • *?/10/2009* – any day in October 2009. • *21/?/2009* – the 21st day of any month in 2009. • *16/?/?* – the 16th day of any month in any year. • *?/7/?* – any day in July in any year. **Note:** Ambiguous searches are not allowed. For example, in a *Month* accuracy search field, *07 ?* is not allowed since it could mean *the 7th of any month* or it could mean *July in any year.* However, the search *July ?* is allowed as it clearly specifies *July of any year.*

8.1.2 Comparison Operators

Range operators in *Whole Number, Currency* and *Date and Time* fields are described in Comparison **Operators: Whole Number, Currency And Date And Time Fields**.

Table 3. Comparison Operators: Whole Number, Currency And Date And Time Fields

Wildcard	Meaning
>	Greater than. Example: • *>1000* – any *Whole Number* or *Currency* value greater than 1000 or any *Year* later than 1000.
<	Less than. Examples: • *<1900* – any *Whole Number* or *Currency* value less than 1900 or any *Year* earlier than 1900. • *< 20 September 2008* – any *Date* before 20th September 2008.
>=	Greater than or equal to. Examples: • *>=1000* – any *Whole Number* or *Currency* value greater than or equal to 1000 or any *Year* greater than or equal to 1000 • *>= 2000* – any *Whole Number* or *Currency* field equal to or greater than 2000 or the Year 2000 or later.
<=	Less than or equal to. Examples: • *<=1900* – any *Whole Number* or *Currency* value equal to, or less than, 1900 or the Year 1900 or earlier. • *<= 20 September 2008* – 20th September 2008 or earlier.

8.1.3 Equality

The equality operator is valid for *All* field types.

Equality can be specified *explicitly* by prefixing the search term with an equals sign. For example, the following two search strings are equivalent:

- *47* – Implicitly expressed: is equal to...
- *=47* – Explicitly expressed: is equal to...

8.1.3.1 Text Fields

If you wish to search for values in a *Text* field beginning with = < > or *!*, you *must* include the = prefix. For example:

- *==a* – searches for =a.
- *=<** – searches for any text beginning with <.
- *=!?* – searches for ! followed by any single character.

8.1.4 Inequality and Inversion

The Inequality and Inversion operators are valid for *All* field types are described in Inequality And Inversion Operators: All Field Types.

Table 4. Inequality And Inversion Operators: All Field Types

Wildcard	Meaning
< >	Not equal to. Examples: • *<>Williams* – not equal to Williams. • *<>100* – not equal to 100.
!	Inversion or Not **Note:** != and <> can often be used interchangeably. Examples • *!=Williams* – not equal to Williams. • *!=20/01/2000* – not equal to 20th January 2000 (assuming a Day accuracy *Date and Time* field and DD MM YYYY Locale) • *!47* – not 47 • *!<47* – not less than 47 • *!20/01/2000* – not 20th January 2000

8.1.5 Negatives

Whole Number and *Currency* search parameter values can contain negative operators to denote numbers less than zero. (There are two negative operators allowed.)

Negative operators in *Whole Number and Currency Fields:*

Table 5. Negative Operators: Whole Number And Currency Fields

Wildcard	Meaning
-	Negative number. Examples: *-1* and *-11.8*
( )	Negative number (Accounting convention) Examples: *(1)* and *(11.8*)

8.1.6 *Date and Time* Keyword Syntax

Date and Time fields support keywords to help simplify searching in Searching Client and when searching a Depot in Management Studio.

Note: All *Keywords* must be prefixed by the *#* character (as shown).

When used in a *Date and Time* search field, the following keywords return the described results.

- *#today* – returns results matching today's date. Can only be used in *Day* and *Time* accuracy *Date and Time* fields.
- *#yesterday* – returns results matching yesterday's date. Can only be used in *Day* and *Time* accuracy *Date and Time* fields.
- *#tomorrow* – returns results matching tomorrow's date. Can only be used in *Day* and *Time* accuracy *Date and Time* fields.
- *#thisyear* – returns results matching the current year. Can be used in *all* accuracy *Date and Time* fields.
- *#thismonth* – returns results matching the current month. Can be used in *Month*, *Day* and *Time* accuracy *Date and Time* fields. (#thismonth *cannot* be used with *Year* accuracy *Date and Time* fields.)
- *#thisweek* – returns results matching any day of the current week. Can only be used in *Day* and *Time* accuracy *Date and Time* fields. By default, weeks start on a Sunday and end on a Saturday. (Con for information on how to change the day of the week on which a week starts.)
- *#lastyear* – returns results matching any day of the previous year. Can be used in *all Date and Time* fields.

- *#lastmonth* – returns results matching any day of the previous month. Can only be used in *Month*, *Day* and *Time* accuracy *Date and Time* field. (#lastmonth *cannot* be used with *Year* accuracy *Date and Time* fields.)
- *#lastweek* – returns results matching any day of the previous week. Can only be used in *Day* and *Time* accuracy *Date and Time* fields. By default, weeks start on a Sunday and end on a Saturday.
- *#q1*, *#q2*, *#q3*, *#q4* – returns results matching any day of the specified *quarter* (that is, 3 month period). Can only be used in *Month*, *Day* and *Time* accuracy *Date and Time* field. (Con for information on how to configure the months making up q1, q2, q3 and q4.)

Note: Quarters apply to the current calendar year. A year cannot be specified with a *quarter* keyword (for example, #q1, #qtr2, #quarter4 etc).

- #qtr1, #qtr2, #qtr3, #qtr4 – same as #q1, #q2. #q3, #q4.
- #quarter1, #quarter2, #quarter3, #quarter4 – same as #q1, #q2. #q3, #q4.
- #now – Current date and time for Time accuracy fields. Current date for Day accuracy fields. Current month for Month accuracy fields. Current year for Year accuracy fields. More Examples of #, below.

Note: The above *Keywords* are *not* case sensitive.

8.1.6.1 More Examples of #now Syntax

#now can also be used with + and - days, weeks and years (depending on the accuracy of the search field).

Examples:

- *#now (+1 day)* – todays's date plus one day. Adding or subtracting a number of days can be used with all *Date and Time* fields.
- *#now (-3 months)* – 3 months before today's date. Adding or subtracting a number of months can be used with month and year accuracy *Date and Time* fields only.
- *#now (+10 years)* – Today's date plus 10 years. Adding or subtracting a number of years can be used with year accuracy *Date and Time* fields only.
- *#now* can also be used with < and >.

Example:

<#now – Any date and time before now.

Note: There cannot be any spaces between the < and *#now.*

#now can also be combined with other search parameters.

Example:

- *>#now (-6 months) <#now* – Any date between now and 6 months ago. This example can be used with month and year accuracy *Date and Time* fields only.

Note: There cannot be any spaces between the > and *#now* or between the < and *#now.*

8.1.6.2 Configuration of Quarters and Weeks

Search users can change the default values for the *#q1, #q2, #q3, #q4* and *#thisweek* search parameter keywords by editing the file *HitecLabs.DataStore.Parsing.dll.config*.

1. To open the configuration file, go to: C:\Dev\DataStore.NET\Common\mainline\Private Assemblies\Parsing\bin (typical folder location)
2. Then open the *HitecLabs.DataStore.Parsing.dll.config* file.
3. Find the following lines:

```
<HitecLabs.DataStore.Parsing.ParserSettings>
    <setting name="StartDayOfWeek" serializeAs="String">
        <value>SUNDAY</value>
    </setting>
    <setting name="StartMonthOfYear" serializeAs="String">
        <value>1</value>
    </setting>
</HitecLabs.DataStore.Parsing.ParserSettings>
```

4. By default, the start month of the year value is 1 (underlined above). 1 is January, therefore quarter one runs from *January to March* of the current year. If you reset the month value to 6 and the current month is 5, the new quarter 1 will run from *June to August* of last year.
5. By default, the start day of the week is *SUNDAY* (underlined above). Therefore the week is configured to run from *Sunday to Saturday*. If you reset the start day of the week value to *MONDAY*, the week will be configured to run from *Monday to Sunday*.

8.1.7 *Date and Time*: Universal Coordinated Time

8.1.7.1 How Date and Time Fields are Stored

Date and Time fields are stored in Zulu Time. Zulu Time is the same as UTC (Universal Coordinated Time) and GMT (Greenwich Mean Time). Therefore, if a document is stored on a system in France (which is 1 hour ahead of Zulu Time) as 1st April 2010 09:00:00 and retrieved in England, the *Item Stored Date and Time* will be displayed as 1st April 2010 08:00:00.

8.1.7.2 Specifying Date and Time Searches In Zulu Time

Date and Time search fields values can be specified in Zulu Time. In the following example, the *Locale* is set to *English (United Kingdom)* and the computer has Daylight saving (or British Summer Time) enabled.

- *hh:mm:ssZ* – specifies the Zulu Time Zone. For example, searching for: *01/08/2012 00:06:00Z* on a system in England during British Summer Time (BST) will retrieve results matching *01/08/2012 01:06:00*. This is because 00:06:00 BST is equivalent to 01:06:00 Zulu Time.
- *hh:mmZ* – the same as *hh:mm:ssZ* above, except *:ss* is not specified.

8.1.8 Combining Search Parameters

For a field, any number of different search parameters can be combined with *[OR]* or *[AND]* operators.

- *[AND]* – retrieves records containing all of the separated expressions
- *[OR]* – retrieves records containing any of the separated expressions. When several values are selected from a Picklist, Searching Client automatically adds the [OR} operator between each picklist value.

Examples:

- *Williams [OR] Williamson* – any result containing either Williams or Williamson.
- *Williams [AND] Williamson* – any result where a search field is linked to multiple Index fields and both the value of *Williams* and the value of *Williamson* are returned.
- *>500 [OR] <100* – any result containing either values greater than 500 or less than 100.
- *>500 [AND] <1000* – any result containing values which are between 501 and 999, inclusive.

Also, a greater than and a less than comparisons can be written together with no operator in between; the appropriate operator is derived according to what is implied.

Examples:

- *<50 >500* – is equivalent to *<50 [OR] >500* i.e. this will return all numbers: up to 49 and above 501, inclusive.
- *>50 <500* – is equivalent to *>50 [AND] <500* i.e. this will return all numbers: from 51 up to 499, inclusive.

Note: The *[OR]* and *[AND]* operators are *not* case sensitive, i.e. *[OR]* is equivalent to *[or]*, *[Or]* and *[oR]*.

8.2 Content Full Text Search Syntax

The following section describes the syntax for *Content Full Text* searching in *Searching Client*.

Note*: Content Full Text* searching only applies to previously configured *Content Full Text* search fields.

When a Content Full Text search field is part of a Search Template with *only* Document-level fields, the Content Full Text search returns Document-level search results. However, if the Search Template contains *only* Page-level fields, the Content Full Text search returns Page-level search results. If the Content Full Text search field is part of a Search Template containing *both* Page-level and Document-level search fields, the Content Full Text search is returned at the appropriate level depending on whether Page-level or Document-level search fields are used in the search. If *no* Document- or Page-level search fields are used in the search, or if *both* Document *and* Page-level fields are used in the search, then Page-level searches are returned.

Searching for two terms in a *Content Full Text* search field is equivalent to searching for the two terms with the *[AND]* operator.

Examples:

- *Mortgage Balance* – Search for results containing both *Mortgage* and *Balance.*
- *Mortgage [AND] Balance* – Search for results containing both *Mortgage* and *Balance.*

Note: All search words must consist of at least two characters

- *Mortgage [OR] Balance* – Search for results containing either *Mortgage* or *Balance* (or both *Mortgage* and *Balance*).

Logical AND binds more tightly than OR, so: *Mortgage Balance [or] Credit* will match documents containing both *Mortgage* and *Balance*, and documents containing *Credit*.

Note: Results matching the Content Full Text search criteria are displayed in the default font colour – that is, you cannot specify a colour for Content Full Text search results.

8.2.1 Quote-delimited Strings

Search strings can be Quote-delimited to indicate the words or phrases which must match the text in the quotes.

Note: *Content Full Text* searches are always case-insensitive. The parameter *Search values are case sensitive* (set in the Search Template) does not apply to *Content Full Text* fields.

Examples:

- *"John Smith"* – searches for results containing John immediately followed by Smith.
- *"John Smith*"* – searches for results containing John immediately followed by a word beginning with Smith. For example, *John Smithson*.

8.2.2 Inequality and Inversion

When used in Content Full Text search fields, the inequality and inversion operators allow you to specify the text you *do not* want to be returned.

Table 6. Inequality And Inversion Operators: Content Full Text Fields

Wildcard	Meaning
! [and] !	Inversion or Not **Note:** *!=* and <> can often be used interchangeably. Examples: • *John !Doe* – containing John but not Doe • *John [and] !Doe* – containing John but not Doe.
<> [and] <>	Not equal to. Examples: • *John <>Doe* – containing John but not Doe • *John [and] <>Doe* – containing John but not Doe.

Note: It is *not* possible to *only* specify unwanted words. At least one wanted word must be included and the wanted word must come first.

8.2.3 Wildcard Character

Wildcard operator in *Content Full Text* Fields: * means one or more characters.

Examples:

- *William** – all words beginning with *William*, for example, *Williams, Williamson*, etc.
- *Will** – all words beginning will *Will*, for example, *Wills, Williams, Williamson,* etc.

Note: A single * at the end of a search term is the *only wildcard supported* by Content Full Text search. It will also work within delimited strings. Quo, above.

8.2.4 Proximity Search

Proximity search can be used with *Content Full Text* fields*. [near]* can be used between two search words to return results where the two words appear within 50 words of each other.

Example:

- *Mortgage [near] Credit* – the word *Mortgage* within 50 words of the word *Credit*.

8.2.5 Ignored Words

The indexing engine does not index every single word in a document – certain commonly used ('noise') words (such as *the*, *on*, *a*) are ignored. The list of words which are not indexed is determined by the language. Therefore if the *Content Full Text* search does not return the expected results, make sure you are not using common words which might be ignored.

8.3 Sub-searches

Sub-searching returns similar results to *Content Full Text* searching. However, as it is a very processor-intensive type of search, it should be used for "one-off" searches only. If this type of search is required on a regular basis for a given Data Definition, *Content Full Text* searching should be configured.

Sub-searching can be performed when it has been enabled in the Search Template in *Management Studio*.

When sub-searching is available, *and* a search value has been entered in regular search field or in a Content Full Text search field, the *sub-search text box is displayed.* Enter a value in the sub-search text box and click *Start Search*.

Sub-search results are displayed in the colour defined in the Style Sheet.

www.hiteclabs.com

430 Bath Road, Slough, Berkshire, SL1 6BB, UK
Tel: +44 (0)1628 600 900 Fax: +44 (0)1628 600 910
Email: sales@hiteclabs.com

945 Concord Street, Framingham, MA 01701-4613, USA
Tel: +1 (508) 620 5372 Fax: + 1 (508) 302 2435
Email: sales@hiteclabs.com